

BASIC MSX
1. METHODES PRATIQUES

JACQUES BOISGONTIER

JACQUES
BOISGONTIER

BASIC MSX

1 - MÉTHODES PRATIQUES



EDITIONS DU P.S.I.
1985

DU MÊME AUTEUR AUX ÉDITIONS DU PSI :

Le basic de A à Z
Le basic et ses fichiers - Tomes 1 et 2
Basic pour tous
Oric pour tous
52 programmes, Oric pour tous
Le cahier du Basic sur Oric
L'Apple et ses fichiers - Tome 1
Apple pour tous
36 programmes, Apple pour tous
Commodore 64 pour tous (avec Gérard Foucault)
Commodore 64, Méthodes pratiques
MO5 et TO7/70 pour tous
MO5 et TO7/70, Méthodes pratiques
Spectrum pour tous (avec Marcel Henrot)

PRÉSENTATION

Vous venez d'acquérir votre nouvel ordinateur au standard MSX et vous désirez découvrir ce qu'il peut vous apporter de plus qu'une machine dotée d'un Basic Microsoft ou Applesoft ?

Destiné à un public déjà initié à la micro-informatique, le "BASIC MSX : Méthodes Pratiques" répondra à votre attente de perfectionnement. Jacques Boisgontier, auteur du best-seller "Le Basic et ses fichiers", y adopte une démarche originale en présentant les instructions du Basic MSX au fur et à mesure de vos besoins.

Si le "BASIC MSX" ne s'adresse pas aux débutants, il comprend cependant en annexe un rappel, sous forme d'initiation, des notions de base nécessaires à la compréhension du texte.

Loin d'être une liste sèche d'instructions, le "Basic MSX" propose quantité de programmes-exemples dans lesquels vous découvrirez de nombreuses astuces utiles.

Cet approfondissement du langage Basic MSX est complété par un chapitre "programmes" où, du jeu au graphisme, en passant par la gestion, vous pourrez mettre en application toutes vos connaissances, et créer de très belles pages-écran. A vous de jouer !

SOMMAIRE

PRISE EN MAIN | 1

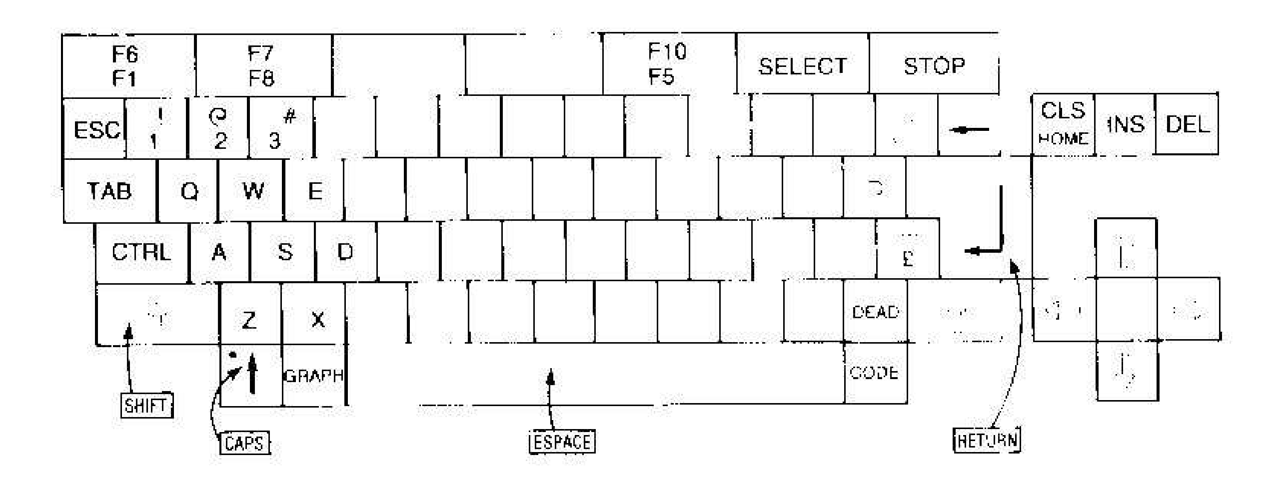
LE CLAVIER

A la mise sous tension, apparaît le message :

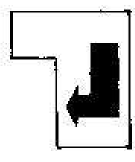
```
MSX BASIC version 1.0
Copyright 1983 by Microsoft
12431 Bytes free
Ok
```

■ ← Curseur

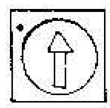
Vous êtes sous BASIC.



Voyons le rôle des touches essentielles :



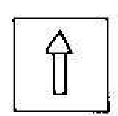
Si vous frappez "PRINT 4 · 5" (afficher 4+5), il ne se passe rien. Pour obtenir un résultat, il faut "valider" la ligne frappee avec la touche ci-contre que nous appelons RETURN.



Lorsque cette touche "CAPS" est enfoncée, les lettres sont affichees en majuscules. Une lampe signale si cette touche est enfoncée ou non.

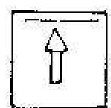


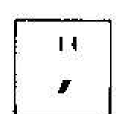
Cette touche, en bas du clavier, déplace le curseur d'une position à droite, sans écrire, permettant ainsi d'introduire des espaces dans le texte frappé.



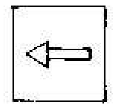
Cette touche, appelée "SHIFT" permet d'accéder aux caractères du haut des touches à deux caractères.
Il faut d'abord appuyer sur cette touche, puis MAINTENIR cette touche tout en appuyant sur le caractère désiré.

Exemple : Pour obtenir le caractère guillemet, appuyez sur les 2 touches ci-dessous SIMULTANÉMENT.





Essayez de frapper : PRINT "BONJOUR".



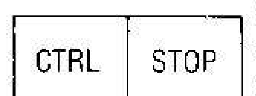
Cette touche permet de déplacer le curseur à gauche et ainsi de modifier un caractère erroné.



Positionne le curseur en haut à gauche de l'écran.



Efface l'écran.



En appuyant **simultanément** sur les touches CTRL et STOP, l'exécution d'un programme est interrompue. Elle peut être poursuivie en frappant "CONT" (cf. chapitre "La mise au point des programmes").

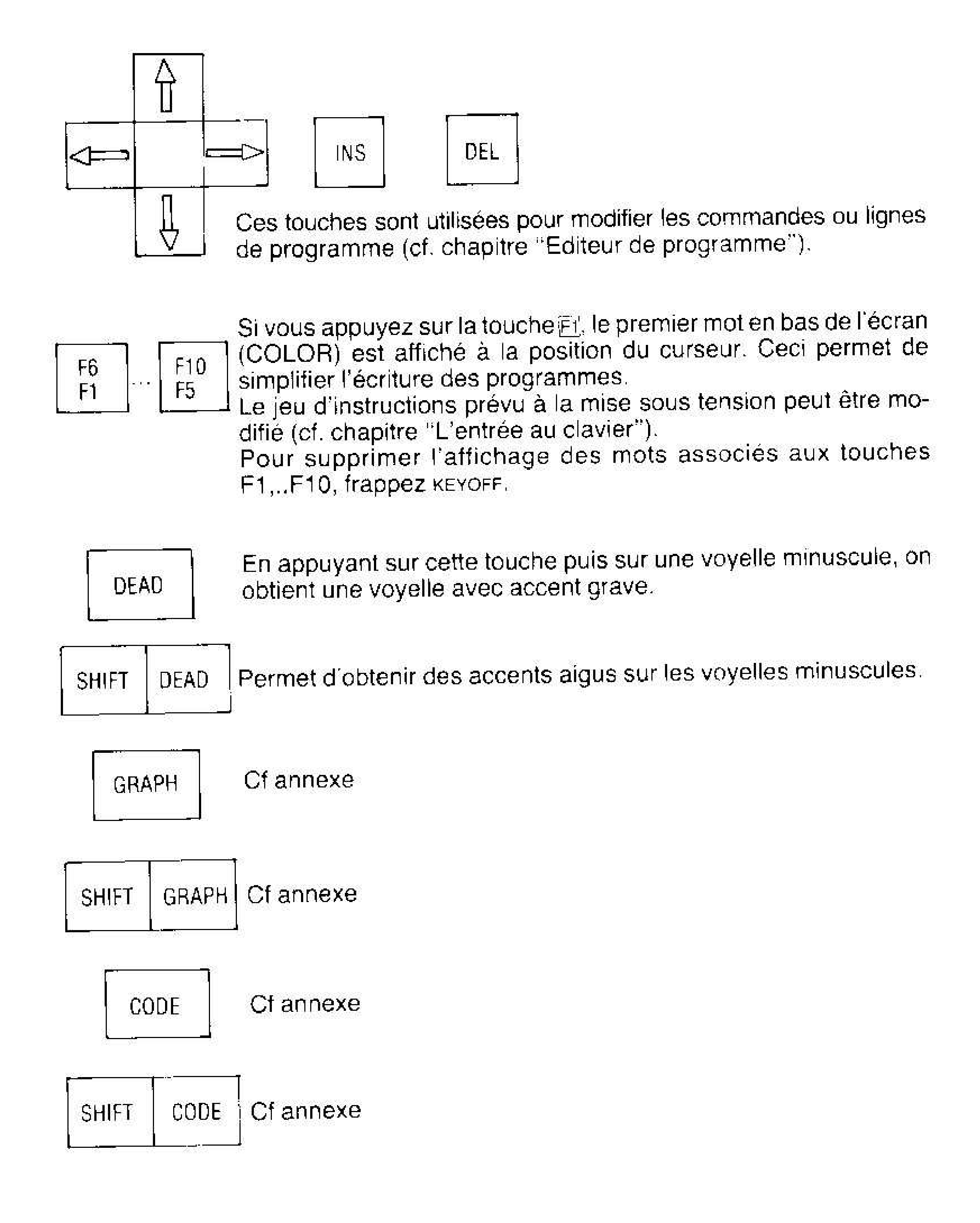
STOP

Suspend l'exécution d'un programme. En appuyant à nouveau sur STOP, l'exécution se poursuit.

Les touches présentées ci-dessous ne sont pas indispensables pour une initiation au BASIC.



Cette touche permet d'accéder aux caractères dont les codes sont compris entre 1 et 31 (cf. annexe).



Ces touches sont utilisées pour modifier les commandes ou lignes de programme (cf. chapitre "Editeur de programme").

Si vous appuyez sur la touche F1, le premier mot en bas de l'écran (COLOR) est affiché à la position du curseur. Ceci permet de simplifier l'écriture des programmes.
Le jeu d'instructions prévu à la mise sous tension peut être modifié (cf. chapitre "L'entrée au clavier").
Pour supprimer l'affichage des mots associés aux touches F1,..F10, frappez KEYOFF.

DEAD : En appuyant sur cette touche puis sur une voyelle minuscule, on obtient une voyelle avec accent grave.

SHIFT DEAD : Permet d'obtenir des accents aigus sur les voyelles minuscules.

GRAPH : Cf annexe

SHIFT GRAPH : Cf annexe

CODE : Cf annexe

SHIFT CODE : Cf annexe

L'ÉDITEUR DE PROGRAMMES

L'éditeur de type "plein écran" est d'un emploi très simple. Il utilise quatre flèches ainsi que les touches INS et DEL.

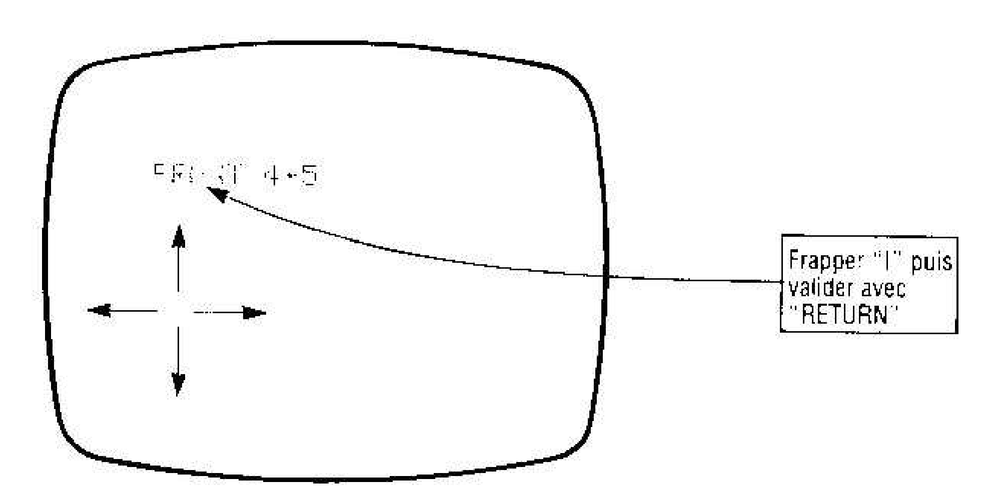
INSERTION D'UNE LIGNE

Pour insérer une ligne entre 10 et 20, on choisit un n° de ligne intermédiaire (15 par exemple).

```
15 PRINT "MONSIEUR"
LIST
10 PRINT "BONJOUR"
15 PRINT "MONSIEUR"
20 GOTO 10
OK
```

MODIFICATION D'UN CARACTÈRE DANS UNE LIGNE DÉJA FRAPPÉE

1/Amener le curseur sur la ligne à modifier à l'aide des flèches ↑ ↓.
2/Positionner le curseur sur le caractère à modifier avec les flèches ← →.
3/Frapper le nouveau caractère qui efface l'ancien.
4/Valider avec la touche RETURN



Pour déplacer le curseur en diagonale, appuyez sur deux flèches simultanément.

SUPPRESSION D'UN CARACTÈRE

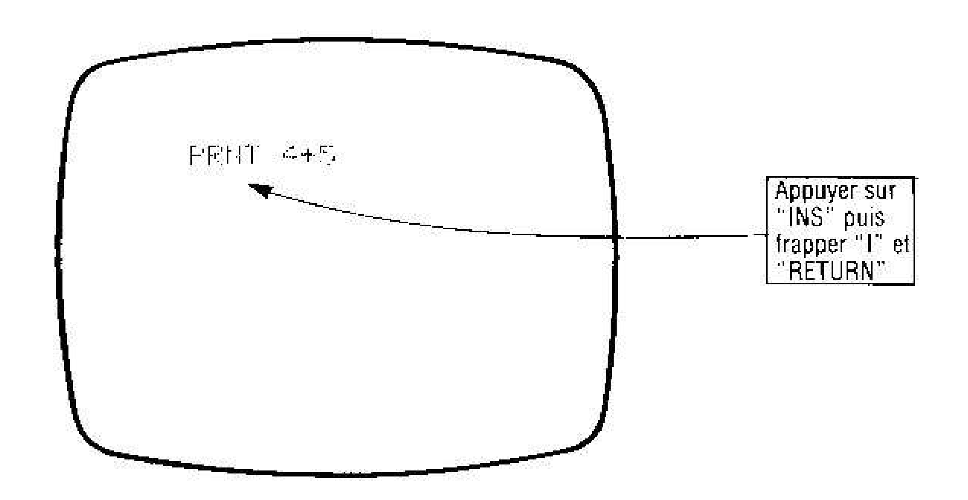
Positionnez le curseur sur le caractère à supprimer. Puis appuyez sur la touche DEL.

INSERTION DE CARACTÈRES

Pour insérer un caractère dans une ligne, on positionne le curseur devant la position d'insertion et on appuie sur INS. Il suffit alors de frapper les caractères à insérer. Les caractères à droite de l'insertion sont décalés automatiquement.
La frappe de RETURN valide la modification.

Une modification en cours peut être annulée en frappant CTRL STOP.

La frappe de l'une des 4 flèches provoque la fin du mode insertion.



DUPLICATION D'UNE LIGNE

Pour dupliquer une ligne de programme (avec un autre numéro), modifiez le numéro de la ligne puis appuyez sur RETURN.
La même ligne de programme existe alors avec 2 numéros.

Attention : Si vous frappez une commande sur une ligne déjà écrite, vous obtenez le message "SYNTAX ERROR".
Dans ce cas, appuyez sur SHIFT CLS et frappez la commande.
(Pour "RUN", appuyez sur SHIFT F10).

Si vous commettez une erreur de frappe en entrant une commande, il n'est pas nécessaire de la frapper à nouveau. Positionnez-vous sur la commande avec les flèches ; corrigez l'erreur puis appuyez sur RETURN.

Exemple :

```
LAST
SYNTAX ERROR
```

Revenez sur "LAST", changez "A" par "I" et appuyez sur RETURN.

Remarque : Lorsque vous ajoutez des caractères à la fin d'une ligne de programme en dépassant une ligne d'écran, il y a ajout de la ligne d'écran suivante. Dans ce cas, utilisez DEL pour supprimer les caractères ajoutés (ou CTRL E).

Les numéros de ligne doivent être compris entre 0 et 65535.
Une ligne de programme ne doit pas excéder 255 caractères.
Le caractère "." référence la dernière ligne utilisée (LIST . par exemple).
Plusieurs instructions sur une même ligne doivent être séparées par le caractère ":".
L'instruction PRINT peut être remplacée par "?".

INSTRUCTIONS DE BASE | 2

LES COMMANDES BASIC

- AUTO
- BLOAD
- BSAVE
- CLEAR
- CLOAD
- CONT
- CSAVE
- DELETE
- LIST
- LOAD
- MERGE
- NEW
- RENUM
- RUN
- SAVE
- TRON
- TROFF

Ces commandes sont acceptées après affichage de "OK". Elles peuvent aussi être écrites comme instructions dans un programme.

AUTO n° ligne départ, incrément

Génère un numéro de ligne à chaque fois que vous appuyez sur RETURN. On sort du mode "AUTO" en appuyant sur CTRL STOP.

```
AUTO          ' numérote de 10 en 10
AUTO 100      ' positionne en 100 et incrémente de 10 en 10
AUTO 100,5    ' positionne en 100 et incrémente de 5 en 5
```

Si un numéro de ligne existe déjà, le caractère "*" est affiché

BLOAD "U:nom-fichier",R,décalage

Charge en mémoire une sauvegarde de la mémoire faite par BSAVE.
"**R**" provoque l'exécution.
"**décalage**" provoque un décalage.

```
BLOAD "CASS:ESS",R
```

BSAVE "U:nom-fichier",adresse début,adresse fin,adresse exécution

Sauvegarde une zone mémoire.

```
BSAVE "CAS:ESS",&H0000,&H1000
```

Sauvegarde sur cassette la zone mémoire comprise entre &H0000 et &H1000.

CLEAR espace chaîne,mémoire maxi

Effacе toutes les variables en mémoire centrale, affecte l'adresse mémoire maximum pour le BASIC et réserve l'espace pour les chaînes qui est de 200 caractères par défaut.

```
CLEAR 1000        ' réserve 1000 caractères pour les chaînes.
CLEAR 1000,60000  ' protège la mémoire au dessus de 60000.
```

CLOAD "nom-programme"

Charge le programme spécifié (sauvegarde par CSAVE). La touche PLAY du lecteur de cassette doit être enfoncée.
Si la télécommande est connectée, le moteur du lecteur est mis en marche automatiquement.

```
CLOAD "ESS"   charge le programme 'ESS'.
CLOAD         charge le premier programme rencontré.
```

Le message "FOUND" est affiché dès que le programme est trouvé.

CLOAD? "nom-programme"

Permet de vérifier qu'une sauvegarde a été effectuée correctement. Le programme en mémoire est comparé à celui sur cassette.

```
CLOAD?
CLOAD? "ESS"
```

CONT

Continue l'exécution d'un programme après arrêt de celui-ci par STOP ou CTRL STOP.
N'est pas accepté si le programme a été modifié (dans ce cas, utiliser GOTO XX).

CSAVE "nom-programme",vitesse

Sauvegarde un programme sur cassette.
"vitesse" égal à 1 spécifie une vitesse de 1200 bauds.
"vitesse" égal à 2 une sauvegarde à 2400 bauds.
Par défaut, la vitesse est de 1200 bauds.
La touche ENREG du lecteur/enregistreur doit être enfoncée.

```
CSAVE "ESS"
CSAVE "ESS",2     ' 2400 bauds
```

La sauvegarde est faite sous forme compactée. Elle est relue par "CLOAD". Si la télécommande est connectée, le moteur du lecteur/enregistreur est mis en marche automatiquement.

DELETE n° ligne début-n° ligne fin

Supprime les lignes entre les limites indiquées.

```
DELETE 100-200   ' supprime les lignes 100 a 200
DELETE .         ' supprime la ligne courante.
```

Les numéros de ligne spécifiés doivent exister.

LIST n° ligne départ-n° ligne fin

Liste les lignes de programme entre les limites indiquées.

```
LIST          ' liste tout le programme
LIST 30-50    ' liste les lignes 30 à 50
LIST 500-     ' liste de 500 à la fin
LIST -400     ' liste jusqu'à 400
LIST .        ' liste la ligne courante
```

LOAD "U:nom-programme",R

Charge un programme sauvegardé en ASCII (par SAVE).
La touche PLAY du lecteur de cassette doit être enfoncée.
"R" provoque l'exécution.

```
LOAD "PAYE"      ' charge le programme 'PAYE'
LOAD "CAS:PAYE"
LOAD "PAYE",R    ' charge le programme et lance l'exécution.
LOAD "CAS:"      ' charge le premier programme rencontré.
```

On ne peut lire par "LOAD" un programme sauvegardé par "CSAVE".
Si la télécommande est connectée, le moteur du lecteur est mis en marche automatiquement.

MERGE "U:nom-programme"

Concatène le programme spécifié (sauvegarde en ASCII par "SAVE") au programme en mémoire centrale. Les lignes de programme dont les numéros sont identiques aux numéros de lignes déjà en mémoire remplacent celles-ci.

Cette commande est généralement utilisée pour ajouter des sous-programmes à un programme.

```
10 MERGE "CAS:HARDC"
20 MERGE "HARDC"
```

NEW

Efface le programme en mémoire centrale.

RENUM nouveau numéro,première ligne,incrément

"nouveau numéro" qui est par défaut égal à 10 spécifie le nouveau premier numéro de ligne.
"première ligne" spécifie la ligne où doit commencer la renumérotation. C'est la première ligne par défaut.
"incrément" représente l'incrément à utiliser pour la renumérotation. Il est égal à 10 par défaut.

```
10 RENUM          ' renumérote de 10 en 10 à partir de 10
20 RENUM 100      ' la première ligne devient 100
30 RENUM 100, 5   ' renumérote de 5 en 5
```

RUN

Exécute le programme en mémoire au premier numéro de ligne.
Toutes les variables sont remises à zéro. Les fichiers sont clos.

RUN n° ligne

Exécute le programme à partir de la ligne spécifiée.

SAVE "U:nom-programme"

Sauvegarde le programme de la mémoire sous forme ASCII, c'est-à-dire sous forme non compactée. Le programme sauvegardé peut être chargé par "LOAD" ou "MERGE".

```
SAVE"CAS:ESSAI"     ' sauvegarde sur cassette
SAVE "ESSAI"
SAVE "CAS:"         ' sauvegarde sur cassette sans donner de nom
```

Sur cassette, la syntaxe **'SAVE" "nom-programme",A'** est équivalente à : **"SAVE "nom-programme"**.

Avec un lecteur de disque, la syntaxe pour une sauvegarde ASCII devient : **'SAVE "nom-programme", A'**.

TRON

Permet de visualiser les numéros des instructions exécutées (cf. chapitre « Mise au point des programmes »).

TROFF

Annule "TRON"
Rappelons que "PRINT FRE(0)" et "PRINT FRE(X$)" donnent respectivement l'espace libre en mémoire et l'espace libre pour les chaînes.

LES VARIABLES

NOMS

Les noms de variables s'écrivent avec plusieurs lettres ou chiffres mais le premier caractère doit être une lettre.
Seuls les deux premiers caractères sont significatifs (PRIX est équivalent à PR).
Un nom de variable ne doit pas comporter un mot clé du BASIC. Ainsi "COIFFEUR" qui contient "IF" n'est pas accepté.

TYPES

Il y a quatre types de variables : les variables **entières, simple-précision, double-précision** et **chaînes de caractères.** Le type est précisé par un caractère à droite de la variable (% $ # !).

Exemple : X% = 5 (variable du type entier).
Par défaut, lorsque le type n'est pas précisé, les variables sont en double-précision.

Type	**Caractère de déclaration**	**Exemple**
Entières (nombres entiers compris entre 32768 et + 32767	%	X% 123: JOURS% = 365
Simple précision (6 chiffres significatifs)	!	SOMME! 1234.56
Double précision 14 chiffres significatifs	#	SOMME# = 123456789.12 #
Chaînes de caractères de longueur variable (255 caractères au maximum)	$	NOM$ = « DUPONT »

Les variables de même nom avec des types différents sont considérées comme des variables distinctes (A et A$ par exemple).
CLEAR et RUN initialisent les variables avec des valeurs nulles (0 pour les variables numériques, chaîne vide pour les variables chaînes).
La place occupée en mémoire va, bien sûr, en croissant avec la précision des nombres. Il en va de même pour les temps de calcul sur ces nombres.
Pour les tables de chaînes, 3 octets par élément seulement sont réservés au moment où "DIM" est exécuté.
Les chaînes elles-mêmes sont rangées dans un espace particulier réservé par "CLEAR ESPACE". Elles utilisent un nombre d'octets égal à leur longueur.

CONSTANTES HEXADÉCIMALES

Des constantes hexadécimales sont spécifiées par "&H" devant le nombre.

```
10 PRINT &HFF  ---> 255
```

Des constantes hexadécimales définies dans des DATA pourront être lues ainsi :

```
10 DATA &HFF
20 READ X:PRINT X
run
 255
```

CONSTANTES OCTALES ET BINAIRES

Des constantes **octales** sont spécifiées par "&O" devant le nombre.
Des constantes **binaires** sont spécifiées par "&B".

Exemple :

```
10 PRINT &O377
20 PRINT &B101
run
 255
 5
```

Les nombres représentés vont de −32768 à +32767. Les nombres négatifs sont représentés sous forme de complément vrai (le premier des 16 bits est positionné à 1 pour les nombres négatifs).

```
octal      décimal hexadécimal

&O100000  -32768   &H8000
&O100001  -32767   &H8001

&O177777  -1       &HFFFF
&O000000  0        &H0000
&O000001  +1       &H0001

&O77777   +32767   &H7FFF
```

DEFINT-DEFSNG - DEFDBL-DEFSTR

(Définition globale de type de variables)

DEFtype lettres — Permet de définir globalement le type de toutes variables dont les noms commencent par les lettres spécifiées, plutôt que de déclarer explicitement le type de chaque variable par un caractère (%, $, #, !).
On peut cependant déclarer explicitement par un caractère (!, %, #, $) un type de variable qui aurait été défini par DEFtype. La déclaration explicite est prioritaire.

DEFINT J — Toutes les variables commençant par J et non déclarées explicitement par un caractère (!, %, $, #) sont des variables entières.

DEFINT A-C — Toutes les variables commençant par une lettre A, B, ou C sont du type chaîne.

DEFINT A-B,D-F — Spécifie deux domaines de variables entières.

```
10 DEFINT J           ' variables commencant par J -> entieres
20 JR=123.456
30 J!=123.456        ' declaration explicite prioritaire
40 PRINT JR,J!

run
 123          123.456
```

CONVERSION DE TYPES DE VARIABLES

Le stockage d'une valeur se fait suivant le type de la variable.

```
10 A%=123.45
20 PRINT A%
run
 123
```

Les opérations sont effectuées en double précision.

```
10 A%=2:B%=3
20 PRINT A%/B%
run
 .6666666666667
```

EFFETS DE LA CONVERSION SUR LA PRÉCISION

La conversion en entier supprime la partie fractionnaire. Un nombre double-précision converti en simple-précision est arrondi.

Remarques : Les variables simple et double-précision sont représentées de façon interne sous forme mantisse/exposant. La mantisse est représentée en Décimal Code Binaire (1 chiffre sur 4 bits). Ceci évite les erreurs d'arrondi qui existent sur certaines versions Microsoft où la représentation est en binaire.

Les valeurs des variables peuvent être comprises entre 10^-61 et 10^61.
Au-delà de 10^{13}, les valeurs sont affichées sous forme mantisse/exposant.

```
10 X=12345678912345#
20 Y=1.2345678912346E+14
30 PRINT X,Y
run
 12345678912345
 1.2345678912346E+14
```

Le type et le nom de la variable occupent 1+2=3 octets.

PLACE OCCUPÉE PAR LES VARIABLES ET TABLES

	variables	tables
entiers	3+2=5	2 octets par élément
simple-précision	3+4=7	4 octets par élément
double-précision	3+8=11	8 octets par élément
chaînes	3+3+ longueur	3+longueur par élément

LES EXPRESSIONS ET OPÉRATEURS

Une expression peut être simplement une constante du type numérique ou chaîne, une variable ou une combinaison de constantes et de variables liées par des opérateurs.

Exemple : (X - 2) + 4/6

Les opérateurs effectuent des opérations sur des valeurs. Ils sont classés en trois catégories :

1/Arithmétiques
2/Relationnels
3/Logiques

OPÉRATEURS ARITHMÉTIQUES

^	Exponentiation
* /	Multiplication Division
+ −	Addition Soustraction
MOD	Modulo
\	Division entière

L'évaluation des expressions se fait avec l'ordre des priorités des opérateurs défini ci-dessus.

5 + 10/5 est égal à 7

Des parenthèses permettent de changer cet ordre. Ce sont d'abord les expressions entre parenthèses qui sont évaluées.

(5 + 10)/5 est égal à 3

X MOD Y : L'opérateur MOD donne le reste de la division de X par Y.

```
PRINT 100 MOD 3 → 1
```

X\Y : Donne la partie entière d'une division. X et Y doivent être compris entre − 32768 et + 32767.

```
100\3 → 33
```

OPÉRATEURS RELATIONNELS

Ils comparent 2 valeurs. Le résultat est soit égal à − 1 (condition vraie), soit égal à 0 (condition fausse). Ceci permet de prendre une décision (par IF... THEN... ELSE).

```
100 IF X<0 THEN PRINT « NOMBRE NEGATIF » ELSE PRINT « NOMBRE POSITIF »
```

=	Egalité
<	Inférieur
>	Supérieur
<>	Différent
<=	Inférieur ou Egal

OPÉRATEURS LOGIQUES

Ils testent des relations multiples. Le résultat de l'opération logique est soit faux (égal à 0) soit vrai (égal à − 1) et permet ainsi de prendre une décision par **IF... THEN... ELSE**.

```
100 IF A<0 AND B<0 THEN PRINT « A et B SONT NEGATIFS »
```

L'ordre des priorités d'évaluation des opérateurs logiques est le suivant :
NOT — AND — OR — XOR — IMP — EQV.

Dans les tables de vérité ci-dessous "0" signifie "FAUX" et "1" signifie "VRAI". Mais en réalité le TEST se fait sur 0 (FAUX) et sur <> 0 (VRAI) et le résultat est soit 0 (faux), soit − 1 (vrai).

R1	R2	
NOT (NON)		NOT R1
1		0
0	—	1

R1	R2	
AND(ET)		R1 AND R2
1	1	1
1	0	0
0	1	0
0	0	0

R1	R2	
OR(OU)		R1 OR R2
1	1	1
1	0	1
0	1	1
0	0	0

R1	R2	
XOR(OU EXCLUSIF)		R1 XOR R2
1	1	0
1	0	1
0	1	1
0	0	0

R1	R2	
IMP(IMPLICATION)		R1 IMP R2
1	1	1
1	0	0
0	1	1
0	0	1

R1	R2	
EQV(EQUIVALENT)		R1 EQV R2
1	1	1
1	0	0
0	1	0
0	0	1

Les priorités d'évaluation des expressions comportant des opérateurs arithmétiques, relationnels et logiques sont :

1/Parenthèses
2/Opérateurs arithmétiques
3/Opérateurs relationnels (1 seul niveau)
4/Opérateurs logiques

```
5 A=7:B=6
10 MXMUM=-((A>B)*A+(A<=B)*B)
15 PRINT MXMUM
```

OPÉRATEURS BOOLEENS

La manipulation de bits et les opérations booléennes sur ces bits s'effectuent avec les opérateurs AND, OR, NOT,...
Ces derniers opèrent sur des groupes de 16 bits au plus qui sont spécifiés par des nombres allant de – 32768 à 32767 (représentés en complément vrai de façon interne).
Les opérations s'effectuent BIT à BIT.

Exemple :
```
15        ———————>0000000000001111
 4        ———————>0000000000000100
15 AND 4  ———————>0000000000000100 - ———>4
```

Exemple :
```
4         ———————>0000000000000100
2         ———————>0000000000000010
2 OR 4    ———————>0000000000000110 ————>6
```

Exemple :
```
– 1       ———————>1111111111111111
  8       ———————>0000000000000000
– 1 AND 8 ———————>0000000000001000 ————>8
```

L'ÉCRAN

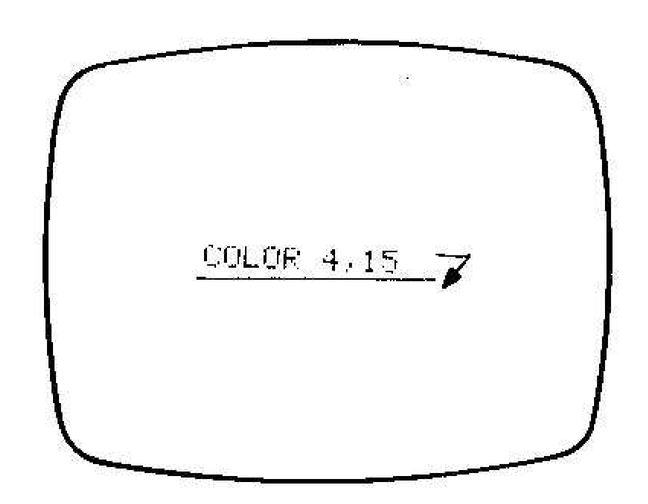
- CLS
- COLOR
- SCREEN
- LOCATE
- BASE

CLS

Efface l'écran.

COLOR écriture,fond,pourtour

Les couleurs d'écriture, de fond et de pourtour se choisissent avec l'instruction "COLOR". A la mise sous tension, l'écriture est blanche sur fond bleu. Si vous frappez "COLOR 4,15", l'affichage se fait en bleu sur fond blanc.



Les couleurs sont définies pour tout l'écran.

```
COLOR 6      ' écriture rouge
COLOR ,15    ' fond blanc
```

Les différentes couleurs sont :

0	transparent	8	rouge
1	noir	9	rouge clair
2	vert	10	jaune foncé
3	vert clair	11	jaune clair
4	bleu foncé	12	vert foncé
5	bleu clair	13	magenta
6	rouge foncé	14	gris
7	cyan	15	blanc

Le programme ci-dessous affiche successivement le texte de l'écran dans les couleurs 1 à 14 sur fond blanc.

```
10 COLOR 1,15
20 PRINT "COULEUR:";
30 FOR CL=1 TO 14
40  COLOR CL
45  PRINT CL;
50  FOR TP=1 TO 1000:NEXT TP
60 NEXT CL
```

Le programme suivant affiche toutes les couleurs en haute résolution.

```
10 '------------- affichage couleurs
20 COLOR 1,15:SCREEN 2
30 '
40 Y=100
50 FOR CL=1 TO 15    ' 15 couleurs
60  X=CL*12
70  LINE (X,Y)-(X+6,Y+16),CL,BF
80 NEXT CL
90 '
100 C$=INPUT$(1)
```

Si un programme utilise par erreur la même couleur pour l'écriture et le fond, l'affichage à l'écran disparaît. Dans ce cas, frappez en mode direct "COLOR 1,15" par exemple pour faire apparaître le texte.

SCREEN mode,type sprite,sono clavier,vitesse cassette,imprimante

■ **Mode :** il existe quatre modes :

0: 24 lignes de 40 caractères
1: 24 lignes de 32 caractères
2: haute résolution 256*192 points
3: basse résolution 64*48 points

Le changement de mode efface l'écran.
Les modes "SCREEN2" et "SCREEN3" ne peuvent être commandés en "mode direct" ; il y aurait retour automatique au mode 0 ou 1.
Le mode "SCREEN0" autorise seulement une couleur de fond et une couleur d'écriture à la fois. Il ne peut y avoir de couleur de bordure ni de sprite en mode "SCREEN0".
En mode "SCREEN2" et "SCREEN3", l'affichage de texte se fait à l'aide de "GRP".
INPUT n'est accepté qu'en mode "SCREEN0" et "SCREEN1".

■ **Type sprite :** quatre types de sprites peuvent être choisis :

0: 8*8 taille simple
1: 8*8 taille double
2: 16*16 taille simple
3: 16*16 taille double

```
10 SCREEN 2,1        ' sprite 8*8 taille simple
20 SCREEN 2,2        ' sprite 8*8 taille double
```

Tous les sprites doivent être du même type.

■ **Sonorisation clavier :**

Egal à zéro, ce paramètre supprime la sonorisation du clavier.

```
10 SCREEN ,,0    ' supprime la sonorisation clavier

10 SCREEN ,,1    ' rétablit la sonorisation clavier
```

■ **Vitesse cassette :**

La vitesse de sauvegarde standard est de 1200 bauds par défaut.
Si le paramètre "vitesse" est égal à 2, la sauvegarde se fait à 2400 bauds. La lecture se fait automatiquement à la vitesse de sauvegarde.

```
10 SCREEN ,,,2        ' cassette 2400 bauds
20 SCREEN ,,,1        ' cassette 1200 bauds
```

■ **Imprimante :**

Une imprimante prévue pour MSX frappe les caractères graphiques MSX. Le paramètre "imprimante" doit être égal à zéro pour ce type d'imprimante.
Pour les imprimantes standards, les symboles graphiques sont remplacés par des espaces.

■ **Jeu de caractères de MSX :**

Le jeu de caractères comprend un jeu standard (32 à 127), un jeu de caractères graphiques (128 à 255) et un jeu de caractères graphiques obtenus par PRINT CHR$(1)+CHR$(X).
En mode "SCREEN0", certains caractères graphiques n'apparaissent pas complètement ; les 2 positions de droite sont supprimées.
On trouvera en annexe la liste des codes de caractères graphiques.

```
10 '-------------- affichage jeu caractères
20 SCREEN 1
30 '
50 FOR I=32 TO 255
60  PRINT CHR$(I);
70 NEXT I
90 '---------- caractères graphiques
95 PRINT
100 FOR I=65 TO 65+31
110  PRINT CHR$(1);CHR$(I);
120 NEXT I
```

```
 !"#$%&'()*+,-./0123456789:;<=
>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^
_`abcdefghijklmnopqrstuvwxyz{|}~
```

LOCATE colonne,ligne,curseur

Positionne le curseur dans la colonne et la ligne spécifiées.
"**colonne**" doit être compris entre 0 et 39 (SCREEN0).
"**ligne**" doit être compris entre 0 et 23.
Les valeurs supérieures aux bornes sont arrondies aux valeurs des bornes.
Ci-dessous, le message "COUCOU" est affiché en colonne 10 et en ligne 5.

```
10 CLS
20 LOCATE 10,5
30 PRINT "COUCOU"
```

Si "curseur" est égal à 0, le curseur n'apparaît pas.

```
10 LOCATE 10,10,0     ' curseur invisible
10 LOCATE 10,10,1     ' curseur visible
```

Le programme ci-dessous fait défiler un message à l'écran.

```
5 '---------- ENSEIGNE
10 CLS
20 X$="LE BASIC MSX......"
30 '
40 LOCATE 5,10:PRINT X$
50 X$=RIGHT$(X$,1)+LEFT$(X$,LEN(X$)-1)
60 FOR TP=1 TO 300:NEXT TP    'temporisation
70 GOTO 40
```

Il n'existe pas de fonction SCRN (col,ligne) donnant le code d'un caractère affiché à l'écran en mode "SCREEN0" ou "SCREEN1".

Cette fonction peut être remplacée par :

VPEEK(AM+C+L*40)

ou "AM" représente l'adresse de la mémoire écran, "C" la colonne et "L" la ligne.

Les programmes ci-dessous recopient sur imprimante le contenu de l'écran.

```
10 '------------ recopie d'écran (screen 0)
20 AM=BASE(0)              ' buffer texte
30 FOR L=0 TO 23           ' 24 lignes
40  FOR C=1 TO 40          ' 40 colonnes
70   CD=VPEEK(AM+C+L*40)'  code
80   LPRINT CHR$(CD);
90  NEXT C
100 LPRINT
110 NEXT L
```

```
10 '------------ recopie d'écran (screen 1)
20 AM=BASE(5)              ' buffer texte
30 FOR L=0 TO 23           ' 24 lignes
40  FOR C=1 TO 32          ' 32 colonnes
70   CD=VPEEK(AM+C+L*32)'  code
80   LPRINT CHR$(CD);
90  NEXT C
100 LPRINT
110 NEXT L
```

BASE(n)

Fournit les adresses de la mémoire écran (cf. chapitre "L'accès à la mémoire").

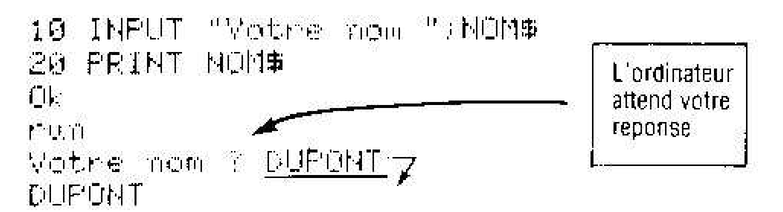
L'ENTRÉE AU CLAVIER

- INPUT
- LINE INPUT
- INPUT$(n)
- INKEY$
- STICK(n)
- ON STRIG GOSUB
- STRIG(n) ON
- STRIG(n) OFF
- STRIG(n) STOP
- PDL(n)
- PAD(n)
- KEY
- KEY LIST
- KEY ON
- KEY OFF
- ON KEY GOSUB
- KEY(n) ON
- KEY(n) OFF
- KEY(n) STOP
- ON STOP GOSUB
- STOP ON
- STOP OFF
- STOP STOP

INPUT "Message" ; variable 1, variable 2,...

Permet d'entrer, pendant l'exécution d'un programme, une ou plusieurs valeurs numériques ou chaînes de caractères. Les variables spécifiées dans INPUT sont séparées par des virgules.
Après l'affichage du message, l'opérateur doit entrer les valeurs des variables dans l'ordre défini par l'instruction INPUT, en les séparant par des virgules.

Exemple avec une variable :



```
10 INPUT "Votre nom ";NOM$
20 PRINT NOM$
Ok
run
Votre nom ? DUPONT
DUPONT
```


Exemple avec deux variables :

```
10 INPUT "Nom,Age ";NOM$,AGE
15 '
20 PRINT NOM$,AGE
Ok
run
Nom,Age ? DUPONT,30
DUPONT         30
```

Si l'opérateur oublie des valeurs, le message "??" est envoyé.

```
list
10 INPUT "Nom,age ";NOM$,AGE
20 PRINT NOM$;AGE
Ok
RUN
Nom,age ? DUPONT
?? 30
DUPONT 30
```

Si l'opérateur appuie sur "RETURN" sans entrer de valeur, une variable conserve son ancienne valeur. Par conséquent, il faut initialiser une variable à zéro avant une instruction "INPUT".

```
10 NOM$="":INPUT "Nom ";NOM$
20 PRINT NOM$
30 GOTO 10
Ok
RUN
Nom ? DUPONT
DUPONT
Nom ?

Nom ? DURAND
DURAND
```

Si l'opérateur entre une chaîne alors que c'est une valeur numérique, le message "REDO" (recommencer) est envoyé.

```
10 INPUT "Age ";AGE
Ok
run
Age ? DUPONT
?Redo from start
Age ? 20
Ok
```

Les chaînes comportant une virgule (séparateur) doivent être entrées entre guillemets.

Exemple :

```
Rue? "11,rue NOBEL"
```

LINE INPUT « MESSAGE » ; variable chaîne

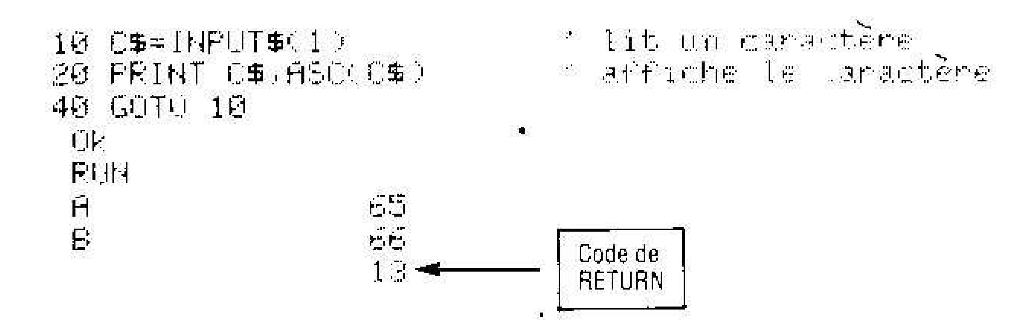
Permet de lire toute une chaîne au clavier, sans tenir compte des séparateurs tels que la virgule comme c'est le cas avec l'instruction INPUT. C'est seulement un retour-chariot qui délimite la fin de la chaîne qui ne peut cependant pas excéder 255 caractères.

```
10 LINE INPUT "Rue? ";RUE$
20 PRINT RUE$
Ok
run
Rue? 11,rue NOBEL
11,rue NOBEL
```

Remarque : BASIC n'envoie pas de point d'interrogation à la suite du message comme c'est le cas avec "INPUT".

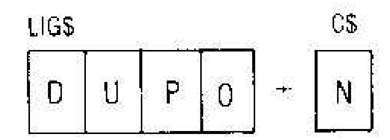
INPUT$(N)

Fournit au programme les N caractères frappés au clavier sans attendre la frappe de "ENTRÉE" (comme pour INPUT).
Ceci permet de mieux contrôler l'introduction des données par l'opérateur.
Les caractères acquis doivent être affichés par le programme.



```
10 C$=INPUT$(1)                    ' lit un caractère
20 PRINT C$,ASC(C$)                ' affiche le caractère
40 GOTO 10
 Ok
 RUN
 A                 65
 B                 66
                   13 <------ Code de RETURN
```

Pour acquérir une chaîne de caractères, il faut concaténer les caractères au fur et à mesure de leur frappe dans une chaîne de caractères (LIG$ sur l'exemple).



Tous les caractères frappés doivent être analysés, y compris le caractère "RETURN" (code ASCII 13) et le caractère "←" (code ASCII 8) (avec "INPUT" ces caractères sont gérés par "BASIC").

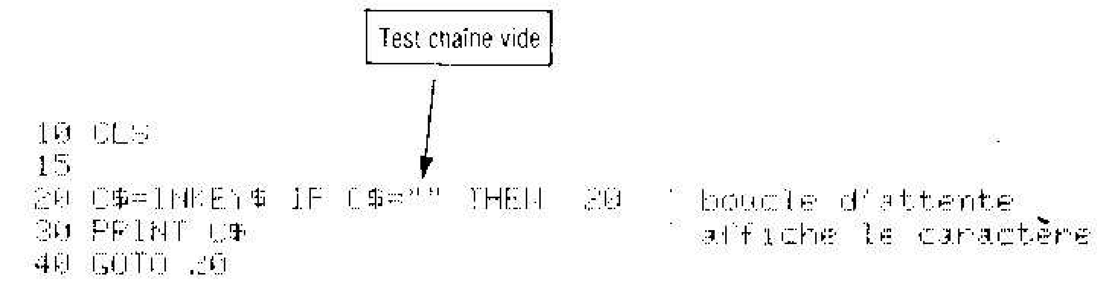
■ Saisie d'une ligne avec INPUT$(1)

```
30 CLS
40 XL=10:YL=10:GOSUB 70          ' coordonnées affichage
50 END
60 '----------------------------
70 LIG$=""
80 '
90 L=LEN(LIG$):LOCATE XL+L,YL
100 '
110 C$=INPUT$(1)
120 C=ASC(C$)
130 '
140 IF C<>8 THEN 170             ' code suppression
150 IF L>0 THEN LIG$=LEFT$(LIG$,L-1):PRINT CHR$(8);
CHR$(32);:GOTO 90 ELSE 90
160 '
170 IF C=13 THEN 220             ' code de return
180 IF C<32 OR C>128 THEN BEEP:GOTO 90
190 LIG$=LIG$+C$                 ' ajout caractère
200 PRINT C$                     ' affichage caractère
210 GOTO 90
220 RETURN
```

INKEY$

L'instruction "INPUT$(1)" est "bloquante" ; si l'opérateur ne frappe pas de caractère au clavier, le programme reste en attente. Il ne peut exécuter d'autres instructions.

INKEY$ lit le clavier en permanence. Si aucun caractère n'a été frappé, la chaîne lue (C$ sur l'exemple) est vide. Le caractère frappé au clavier n'est affiché que si le programme l'a prévu (et non pas automatiquement comme c'est le cas avec INPUT).




```
10 CLS
15
20 C$=INKEY$:IF C$="" THEN 20   ' boucle d'attente
30 PRINT C$                     ' affiche le caractère
40 GOTO 20
```

Remarque : Avec INKEY$, le curseur n'apparaît pas.

Ci-dessous, tant que vous n'appuyez pas sur une touche, le message "APPUYEZ SUR UNE TOUCHE" est affiché.

```
10 C$=INKEY$
15 IF C$<>"" THEN END
20 PRINT "APPUYEZ SUR UNE TOUCHE"
30 GOTO 10
Ok
RUN
APPUYEZ SUR UNE TOUCHE
APPUYEZ SUR UNE TOUCHE
APPUYEZ SUR UNE TOUCHE
APPUYEZ SUR UNE TOUCHE
```

BOUCLE D'ATTENTE

Attend que l'opérateur appuie sur une touche quelconque.

```
10 X$=INKEY$:IF X$="" THEN 10     ' boucle d'attente
20 PRINT "C'EST PARTI"
```

Teste si l'opérateur répond assez vite.

```
10 PRINT "REPONDEZ O/N (VITE) ";
20 '
30 TIME=0
40 '
50 R$=INKEY$:IF R$<>"" THEN 90
60 IF TIME/50 >4 THEN PRINT "TROP TARD":END
70 GOTO 50
80 '
90 ' suite
```

STICK(0)

Permet de lire le clavier comme un joystick. Les 8 directions sont obtenues avec les quatre flèches (les directions diagonales sont obtenues en appuyant sur deux flèches simultanément).
Les déplacements obtenus sont plus rapides qu'avec "INKEY$" (il n'y a pas de délai pour la répétition).
Naturellement, c'est dans les programmes d'animation que cette fonction est surtout utilisée.
STICK(0) égal à zéro indique qu'aucune des flèches n'est enfoncée.

```
10 '---------- TELECRAN AVEC STICK
20 '
30 CF=15:CL=2                         ' couleur fond et écriture
40 COLOR CL,CF
50 SCREEN 2
60 '
70 OPEN "GRP:" FOR OUTPUT AS #1
80 DRAW "BM10,170"
90 PRINT #1,"FLECHES "
100 X=100:Y=100                       ' coordonnées départ
110 '--------------- curseur clignotant
120 C=STICK(0):IF C<>0 THEN 170 ' test clavier
130 '
140 PSET(X,Y),CL:PSET(X,Y),CF
150 GOTO 120
160 '-------
170 PSET(X,Y),CL
180 '
190 IF C=7 THEN X=X-1                 ' gauche
200 IF C=3 THEN X=X+1                 ' droite
210 IF C=5 THEN Y=Y+1                 ' bas
220 IF C=1 THEN Y=Y-1                 ' haut
230 IF C=8 THEN X=X-1:Y=Y-1           ' gauche/haut
240 IF C=2 THEN X=X+1:Y=Y-1
250 IF C=6 THEN X=X-1:Y=Y+1
260 IF C=4 THEN X=X+1:Y=Y+1
270 '
280 IF X<1 THEN X=1
290 IF X>254 THEN X=254
300 IF Y<1 THEN Y=1
310 IF Y>180 THEN Y=180
320 GOTO 120
```

STICK(1 ou 2)

Cette fonction donne la direction (parmi 8) du manche d'un joystick connecté sur l'entrée 1 ou 2. Les règles d'utilisation sont les mêmes que pour STICK(0).

ON STRIG GOSUB n° ligne0,n° ligne1,..,n° ligne4
STRIG(n) ON
STRIG(n) OFF
STRIG(n) STOP

ON STRIG GOSUB définit les numéros de ligne vers lesquels il y aura déroutement du programme si l'opérateur appuie sur la barre "ESPACE" ou les boutons de validation des joysticks.

n° ligne 0 : barre espace
n° ligne 1 et 3 : joystick 1
n° ligne 2 et 4 : joystick 2

```
10 ON STRIG GOSUB 100      ' clavier
10 ON STRIG GOSUB ,200,,400   ' joystick 1
```

STRIG(n) ON valide la déclaration faite par ON STRIG GOSUB.
STRIG(n) OFF annule la validation.
STRIG(n) STOP suspend la validation et mémorise l'action sur la barre "ESPACE" ou sur les boutons de validation en attendant ON STRIG(n) ON.

n=0 : barre espace
n=1,3 : joystick1
n=2,4 : joystick2

```
10 ON STRIG GOSUB 100
20 STRIG(0) ON
30 '
40 PRINT "APPUYEZ SUR ESPACE"
50 GOTO 40
60 '
90 '--------------- sous-Programme ON STRIG
100 PRINT "JE SUIS PASSE PAR LA"
110 RETURN

APPUYEZ SUR ESPACE
APPUYEZ SUR ESPACE
APPUYEZ SUR ESPACE
JE SUIS PASSE PAR LA
APPUYEZ SUR ESPACE
APPUYEZ SUR ESPACE
```

STRIG(n)

Teste si la barre "ESPACE" ou les boutons de validation des joysticks 1 et 2 sont appuyés.

n=0 : barre espace
n=1,3 : joystick 1
n=2,4 : joystick 2

```
10 X=STRIG(0)
20 IF X=-1 THEN PRINT "OUI" ELSE PRINT "NON"
30 GOTO 10
```

PAD(n)

Lit des valeurs fournies par des périphériques du type "tablette graphique" connectés sur les entrées des joysticks 1 et 2.

- joystick1 : PAD(0) : −1 si valeur disponible
  PAD(1) et PAD(2) : X et Y
  PAD(3) : bouton de validation

■ joystick2 : PAD(4) : −1 si valeur disponible
PAD(5) et PAD(6) : X et Y
PAD(7) : bouton de validation

```
10 IF PAD(0)<>-1 THEN 10  ' boucle d'attente
20
100 X=PAD(1):Y=PAD(2)
110 PRINT X,Y
120 GOTO 10
```

Le programme ci-dessous trace un dessin par segments de droite à partir du point 100,100.

```
10 SCREEN 2
20 PSET(100,100)
30 '
40 IF PAD(0)<>-1 THEN 40    ' périphérique prêt
50 IF PAD(3)<>-1 THEN 40    ' validation?
60 '
70 X=PAD(1):Y=PAD(2)
80 LINE -(X,Y)
90 GOTO 40
```

PDL(n)

Lit des valeurs X et Y sur une palette MSX connectées sur les entrées des joysticks 1 et 2. Si rien n'est connecté, la valeur lue est 255.

"n" doit être compris entre 1 et 12.
n = 1,3,5,7, 9,11 → joystick 1
n = 2,4,6,8,10,12 → joystick 2

```
10 X=PDL(1):Y=PDL(2)
20 PRINT X,Y
30 GOTO 10
```

KEY numéro touche,chaîne

Affecte une chaîne, représentant une instruction ou commande, à une touche de fonction (F1,F2,..F10).
En frappant une touche de fonction, la chaîne associée est affichée à l'écran. Si la chaîne est suivie d'un retour-chariot, l'instruction ou commande est exécutée automatiquement.

```
10 KEY 1,"RUN"+CHR$(13)

10 KEY 2,"LIST"
```

KEY LIST

Fournit la liste des instructions associées aux touches de fonction.

```
key list     color 15,4,7
color        cload"
auto         cont
goto         list.
list         run
run
```

KEY OFF

Supprime l'affichage, en bas de l'écran, des chaînes associées aux touches de fonction.

KEY ON

Provoque l'affichage, en bas de l'écran, des chaînes associées aux touches de fonction.

ON KEY GOSUB n° ligne1,n° ligne2,..,n° ligne10
KEY(n) ON
KEY(n) OFF
KEY(n) STOP

ON KEY GOSUB définit les numéros de ligne vers lesquels il y a débranchement si l'opérateur appuie sur une touche de fonction pendant l'exécution d'un programme.
KEY(n) ON valide la déclaration faite par ON KEY GOSUB.
KEY(n) OFF annule la validation faite par KEY(n) ON.
KEY(n) STOP suspend la validation et mémorise l'action sur une touche de fonction en attendant KEY(n) ON.

```
10 ON KEY GOSUB 90,120,150
15 '
20 KEY(1) ON          ' validation F1
30 KEY(2) ON          ' validation F2
40 KEY(3) ON          ' validation F3
50 '
60 PRINT "APPUYEZ SUR F1,F2,F3"
70 GOTO 60
80 '
90 PRINT "FONCTION 1"
100 RETURN
110 '
120 PRINT "FONCTION 2"
130 RETURN
140 '
150 PRINT "FONCTION 3"
160 RETURN
```

```
APPUYEZ SUR F1,F2,F3
APPUYEZ SUR F1,F2,F3
FONCTION 1
APPUYEZ SUR F1,F2,F3
APPUYEZ SUR F1,F2,F3
APPUYEZ SUR F1,F2,F3
FONCTION 1
APPUYEZ SUR F1,F2,F3
APPUYEZ SUR F1,F2,F3
```

ON STOP GOSUB n° ligne
STOP ON
STOP OFF
STOP STOP

ON STOP GOSUB définit le numéro de ligne vers lequel il y a débranchement si l'opérateur appuie sur "CTRL/STOP".
STOP ON valide la déclaration faite par ON STOP GOSUB.
STOP OFF annule la validation faite par STOP ON.
STOP STOP suspend la validation et mémorise l'action sur "CTRL/STOP" en attendant STOP ON.

Le retour s'effectue après l'instruction interrompue (INPUT sur l'exemple).

```
10 ON STOP GOSUB 110          ' numero de ligne si CTRL/STOP
20 '
30 STOP ON                    ' validation ON STOP
40 '
50 R$="":INPUT "Repondez(O/N) ";R$
60 PRINT "COUCOU"
70 IF R$=""THEN 50
80 '
90 GOTO 50
100 '------------------- sous-Programme ON STOP
110 INPUT "On arrete vraiment(O/N) ";RP$
120 IF RP$="O" THEN END
130 RETURN
```

TESTS

- IF...THEN...ELSE
  SI...ALORS...SINON

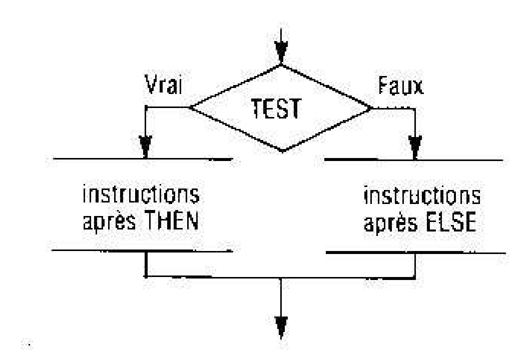
IF...THEN...ELSE

IF expression logique — vraie ou fausse

THEN suite d'instructions — si expression logique vraie

ELSE suite d'instructions — si expression logique fausse

Cette instruction teste si une expression logique est vraie ou fausse.
Si celle-ci est vraie, alors toutes les instructions comprises entre **THEN** et **ELSE**, sont exécutées.
Sinon, ce sont toutes les instructions après le **ELSE** qui sont exécutées.




```
10 INPUT "Nombre A,Nombre B ";A,B
20 IF A>B THEN PRINT "A > B" ELSE PRINT "A <=B"
30 PRINT "suite"
40 GOTO 10
 Ok
 run
 Nombre A,Nombre B ? 3,5
 A <=B
 suite
 Nombre A,Nombre B ? 6,2
 A > B
 suite
 Nombre A,Nombre B ?
```

En fait, le test peut se faire, non seulement sur une expression logique, mais aussi sur une expression arithmétique qui est interprétée comme fausse si elle a une valeur nulle ou comme vraie pour toute autre valeur. Mais on évitera d'utiliser cette particularité du BASIC.

```
5 I=4
10 IF I THEN PRINT "I est different de 0"
```

```
5 I=12
10 IF (I>10)*(I<20) THEN PRINT "I est compris entre 10 et 20"
```

Les IF...THEN...ELSE... peuvent être emboîtés, mais il faut alors bien s'assurer qu'à chaque IF-THEN il correspond un ELSE.

```
10 INPUT "Nombre A,Nombre B ";A,B
20 IF A<=B THEN IF A<B THEN PRINT "A<b" ELSE PRINT "A=B" ELSE
PRINT "A>B"
```

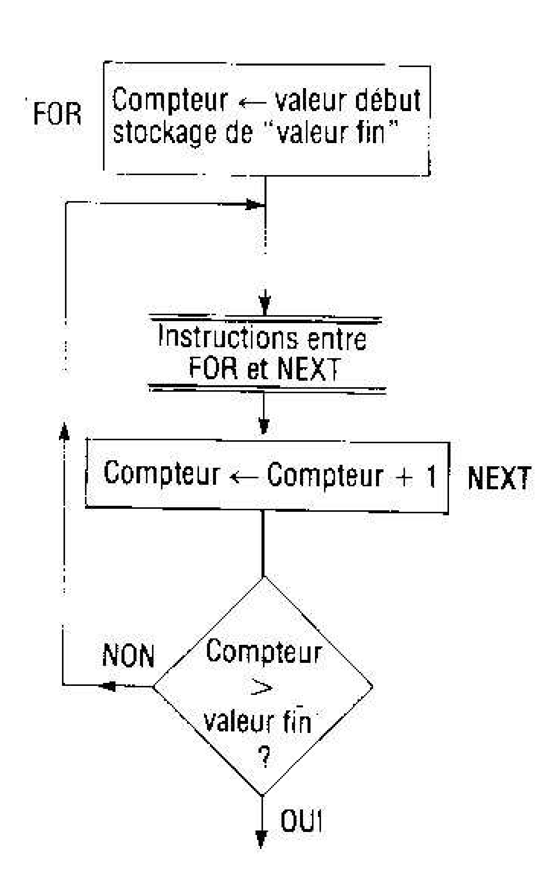
BOUCLE AUTOMATIQUE

- FOR
- NEXT

FOR variable-compteur = valeur début TO valeur fin STEP pas
- **instruction 1**
- **instruction 2**

NEXT variable compteur

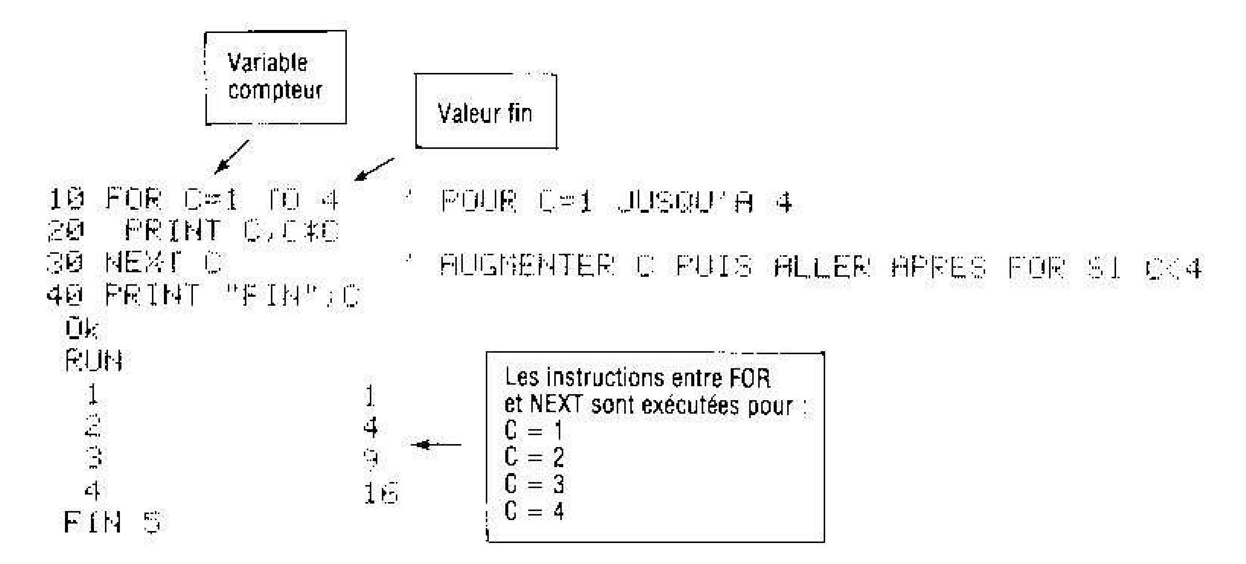
Permet d'écrire des boucles de façon plus concise.
Lorsque l'instruction FOR est exécutée, le BASIC MSX affecte à la "variable-compteur" la "valeur début" spécifiée et mémorise la "valeur fin" indiquée après TO.
Toutes les instructions entre FOR et NEXT sont d'abord exécutées avec "variable-compteur"="valeur début". L'exécution de NEXT augmente la valeur de la "variable-compteur" du pas spécifié dans STEP (1 par défaut).
Si la valeur de "variable-compteur" est inférieure ou égale (pour un STEP positif) à "valeur fin", l'exécution se poursuit à l'instruction après FOR. Par conséquent les instructions entre FOR et NEXT sont à nouveau exécutées avec la nouvelle valeur de la variable-compteur.
Si la valeur de la **"variable-compteur" dépasse "valeur fin"** (pour un pas positif), **l'exécution de la boucle s'achève et le programme se poursuit après NEXT.**



A la fin de la boucle, la **"variable-compteur" a une valeur égale à "valeur fin" + 1.**

Exemple :

Le programme ci-dessous affiche les carrés des nombres 1 à 4



Exemple avec STEP négatif :

```
10 FOR C=4 TO 1 STEP-1
20  PRINT C,C*C
30 NEXT C
40 PRINT "FIN";C
 Ok
 RUN
  4             16
  3             9
  2             4
  1             1
 FIN 0
```

Calculées au moment de l'exécution de l'instruction FOR, donc une seule fois, "valeur début","valeur fin" et "pas" ne varient pas en cours d'exécution de la boucle si les valeurs des variables qui ont servi à les calculer évoluent pendant l'exécution. En revanche, la valeur de la variable-compteur peut être modifiée.

Ne peut être modifié pendant l'exécution

```
10 F=1
20 X=10
30 FOR I=1 TO X
40  IF I=5 AND F=1 THEN X=8
50  PRINT I;
60 NEXT I
Ok
RUN
 1  2  3  4  5  6  7  8  9  10
```

→

```
10 FOR I=1 TO 5
20  PRINT I;
30  INPUT"NOMBRE ";A(I)
40  IF A(I)<0 THEN PRINT "ERREUR": I=I-1
50 NEXT I
Ok
RUN
 1 NOMBRE ? 4
 2 NOMBRE ? -2
ERREUR
 2 NOMBRE ? 3
 3 NOMBRE ? 5
 4 NOMBRE
```

On revient en arrière

C'est en fin de boucle que se fait la comparaison de la valeur de la variable-compteur à la valeur finale. La boucle est donc exécutée une fois si la valeur finale est inférieure à la valeur initiale (pour un incrément positif).

```
10 D=5:F=2
20 FOR I=D TO F
30  PRINT I
40 NEXT I
Ok
RUN
 5
```

Pour éviter cela, il faut ajouter un test :

25 IF F<D THEN 40

Si la "variable-compteur" est du type entier, l'exécution de la boucle est plus rapide.

BOUCLES EMBOÎTÉES

Plusieurs boucles FOR... NEXT peuvent être "emboîtées", c'est-à-dire qu'une boucle peut être placée à l'intérieur d'une autre. Mais il est interdit de les faire se chevaucher.

Exemple :

```
10 FOR I=1 TO 3              ' boucle externe
20  PRINT "I=";I;"J=";
30  FOR J=1 TO 8             ' boucle interne
40   PRINT J;
50  NEXT J
60  PRINT
70 NEXT I
Ok
run
 I= 1 J= 1  2  3  4  5  6  7  8
 I= 2 J= 1  2  3  4  5  6  7  8
 I= 3 J= 1  2  3  4  5  6  7  8
```

S'il n'y a pas d'instruction entre NEXT J et NEXT I, NEXT J,I donne le même résultat.
NEXT (au lieu de NEXT J) est accepté, puisqu'en fait NEXT incrémente le compteur du FOR le plus récent et que celui-ci est supprime dès qu'il atteint la valeur limite. Mais pour des raisons de lisibilité, on indiquera le nom de la variable.

SORTIE D'UNE BOUCLE FOR

On peut sortir d'une boucle FOR par "GOTO" **sans problème**. (Les versions Microsoft antérieures posaient un problème dans le cas où un indice non "épuisé" était utilisé dans une autre boucle FOR "interne".)

```
10 FOR I=1 TO 5
20  IF I=3 THEN GOTO 50       ' sortie par GOTO
30 NEXT I
40 '
50 FOR J=1 TO 3
60  FOR I=1 TO 4              ' I utilise a nouveau
70   PRINT "I=";I,
80  NEXT I
85  PRINT
90 NEXT J
100 PRINT FRE(0)
110 GOTO 10
 Ok
 run
 I= 1 I= 2 I= 3 I= 4
 I= 1 I= 2 I= 3 I= 4
 I= 1 I= 2 I= 3 I= 4
  12034
 I= 1 I= 2 I= 3 I= 4
 I= 1 I= 2 I= 3 I= 4
 I= 1 I= 2 I= 3 I= 4
  12034
```

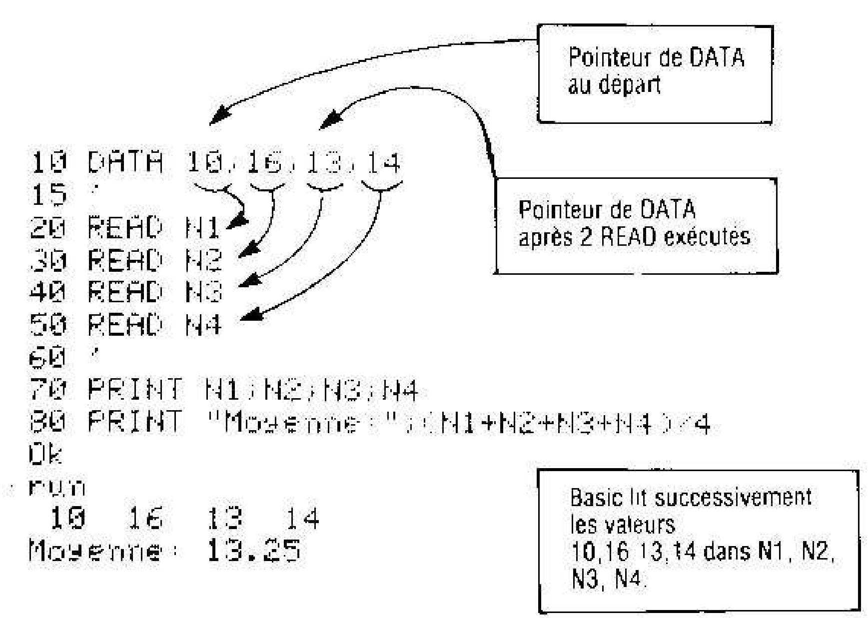
TRAITEMENT DES DONNÉES | 3

LES DONNÉES

- DATA
- READ
- RESTORE

DATA

L'instruction DATA permet de définir des données dans le programme lui-même. Celles-ci sont ensuite lues dans des variables par l'instruction "READ VARIABLE" (LIRE variable).



READ

"READ N1" lit la 1re donnée (10) dans N1.
Le pointeur de DATA (géré par BASIC) progresse de 1. Ainsi "READ N2" lit la 2e donnée dans N2, etc...

Les données peuvent être écrites sur plusieurs lignes :

```
10 DATA 10,16
20 DATA 13,14
```

Les lignes 10 et 20 sont équivalentes à la ligne :

```
10 DATA 10,16,13,14
```

L'implantation des DATAS dans un programme n'a pas d'importance. Elles sont lues dans l'ordre de la numérotation.

Les chaînes de caractère comportant des caractères spéciaux doivent être placées entre guillemets.

Sur l'exemple ci-dessous, sans la présence de guillemets, la virgule serait considérée comme séparateur.

```
10 DATA "12,rue LAGAFFE"
20 '
30 READ X$
40 PRINT X$
Ok
run
12,rue LAGAFFE
```

RESTORE

Positionne en début de DATA ce qui permet de relire les données depuis le début.

```
10 DATA 6,3,14
20 '
30 READ A,B,C          ' 1ere lecture
40 '
50 RESTORE             ' debut DATA
60 '
70 READ D,E,F          ' 2eme lecture
80 '
85  PRINT
90 PRINT A;B;C
100 PRINT D;E;F

run

 6  3  14
 6  3  14
```

RESTORE n° ligne

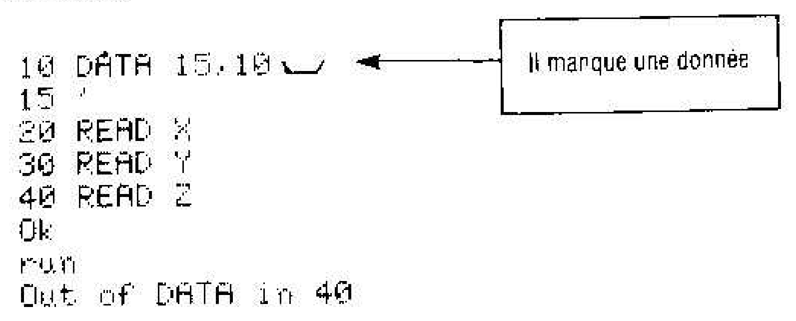
Positionne sur le numéro de ligne de DATA spécifié, vous permettant ainsi de sélectionner des données particulières.

```
10 DATA 6,3,14
20 DATA 4,2,8
30 RESTORE 20          ' ligne 20
40 READ X,Y,Z
50 PRINT X;Y;Z
Ok
run
 4  2  8
```

ERREURS

■ OUT OF DATA :

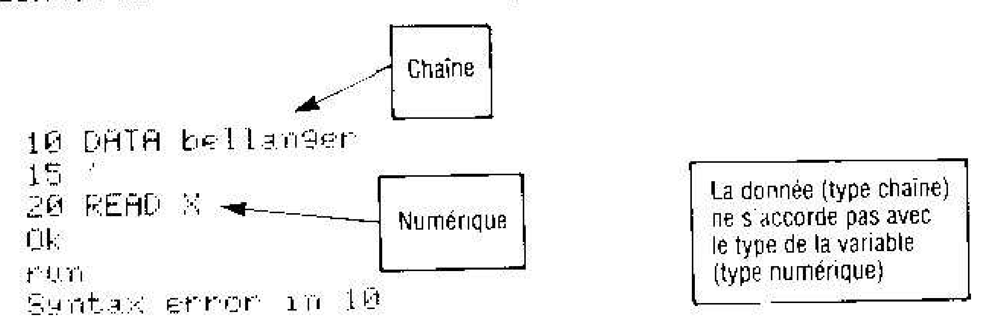
Si le nombre de READ exécutés est supérieur au nombre de données en DATA, le message OUT OF DATA est affiché.




```
10 DATA 15,10
15 '
20 READ X
30 READ Y
40 READ Z
Ok
run
Out of DATA in 40
```

■ SYNTAX ERROR :

Le type de la donnée lue doit s'accorder avec le type de la variable.




```
10 DATA bellanger
15 '
20 READ X
Ok
run
Syntax error in 10
```

L'exemple ci-dessous lit un nom au hasard dans une liste de 4 noms.

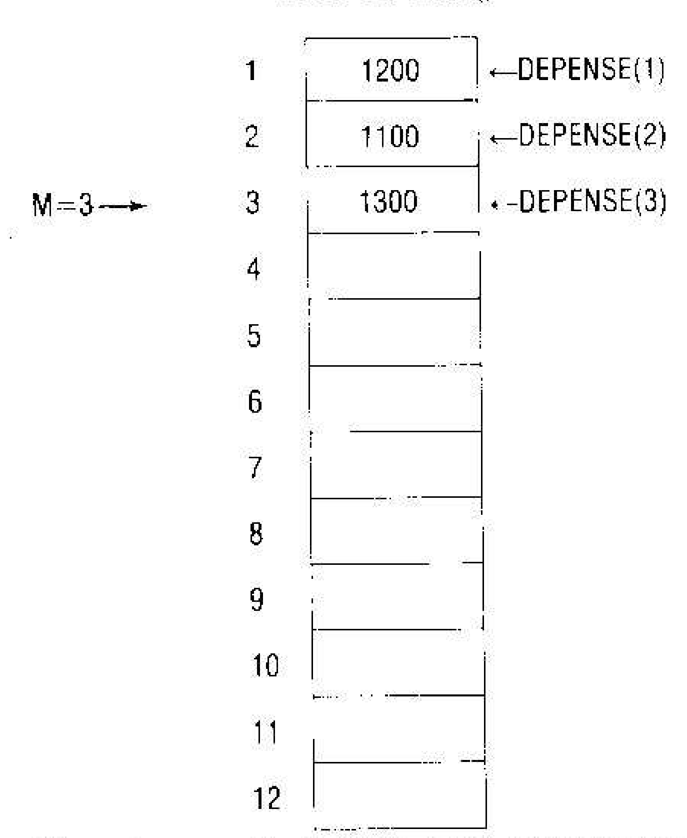
```
10 DATA Pierre,Paul,Jacques,nicolas
20 '
30 '
40 RESTORE
50 X=INT(RND(1)*4)
60 '
70 IF X=0 THEN 100
80 FOR I=1 TO X:READ X$:NEXT I
90 '
100 READ NOM$
110 PRINT NOM$
120 GOTO 40
 Ok
 run
 Jacques
 Pierre
 nicolas
```

Ci-dessous, nous sélectionnons un groupe de DATA. Chaque groupe peut comporter plusieurs lignes. La fin de chaque groupe est repérée par le caractère "*".

```
10 DATA maison ,porte,chambre
20 DATA *
30 DATA jean,pierre,paul,jacques
40 DATA *
50 DATA roue,cadre,frein,pedale,selle
60 DATA *
70 '
80 RESTORE
90 INPUT "Quel groupe (1,2,3) ";G
100 '
110 IF G=1 THEN 170
120 '
130 '
140 READ X$:IF X$="*" THEN G=G-1:GOTO 110
150 GOTO 140
160 '
170 READ MOT$                    ' 1er data du groupe
180 PRINT MOT$
190 GOTO 80
Ok
run
Quel groupe (1,2,3) ? 2
jean
Quel groupe (1,2,3) ? 1
maison
```

LES TABLES

Les tables contiennent des éléments de même nature auxquels nous accédons par un indice. Nous pouvons ainsi traiter les éléments d'une table en faisant simplement varier un indice.
Une table comportant plus de 10 éléments doit être dimensionnée par **DIM nom-table (nombre éléments)**
Soit une table des dépenses relatives aux 12 mois de l'année. Les éléments de cette table que nous appelons DEPENSE() sont connus sous les noms de **DEPENSE(1),DEPENSE(2),...,DEPENSE(12).**



Pour documenter la table, nous faisons varier un indice M de 1 à 12.

```
10 DIM DEPENSE(12)
20 '
30 FOR M=1 TO 12
40  PRINT "Mois";M;
50  INPUT DEPENSE(M)
60 NEXT M
 Mois 1 ? 1200
 Mois 2 ? 1100
 Mois 3 ? 1300
 Mois 4 ? 700
 Mois 5 ? 800
 Mois 6 ? 900
 Mois 7 ? 1000
 Mois 8 ? 1200
 Mois 9 ? 1100
 Mois 10 ? 1400
 Mois 11 ? 800
 Mois 12 ? 1200
```

Au départ, l'indice M étant égal à 1, l'instruction INPUT DEPENSE(M) est équivalente à INPUT DEPENSE(1).
Par conséquent, c'est dans l'élément DEPENSE(1) que la première valeur est introduite. Au second passage dans la boucle, M étant égal à 2, c'est dans DEPENSE(2) que la seconde valeur est introduite. Etc.
Pour connaître le total des dépenses entre 2 mois M1 et M2, nous écrivons :

```
10 DIM DEPENSE(12)
20 '
30 FOR M=1 TO 12
40  PRINT "Mois";M;
50  INPUT DEPENSE(M)
60 NEXT M
70 '-----------------
90 PRINT
100 INPUT "Mois 1";M1
110 INPUT "Mois 2";M2
120 '
140 TTAL=0
150 FOR M=M1 TO M2
160  TTAL=TTAL+DEPENSE(M)
170 NEXT M
180 PRINT
190 PRINT "Total:";M1;M2;TTAL
200 GOTO 90

run
 Mois 1? 1
 Mois 2? 12

 Total: 1  12  12700

 Mois 1? 1
 Mois 2? 6

 Total: 1  6  6000
```

Il existe pour les tables un élément 0 :

100 A(0)= 15

TRI DES ÉLÉMENTS D'UNE TABLE

Rangeons dans l'ordre croissant les éléments d'une table. Pour cela, nous allons utiliser l'algorithme suivant. Nous comparons d'abord le deuxième élément de la table au premier :

- **s'il est plus grand**, nous les laissons dans l'ordre
- **s'il est plus petit**, nous les inversons afin qu'ils soient dans l'ordre.

```
120 IF A(I+1)<A(I) THEN   SWAP A(I+1),A(I)  INV=1
     !                !       !
    SI A(I+1)<A(I) ALORS  ECHANGER A(I+1),A(I):INV=1
```

Ensuite, nous progressons de 1 pas en comparant de la même façon le troisième élément au deuxième, et nous les inversons s'ils ne sont pas dans l'ordre.
etc.
Lorsque nous arrivons à la fin de la table, nous regardons s'il y a eu ou non inversion :

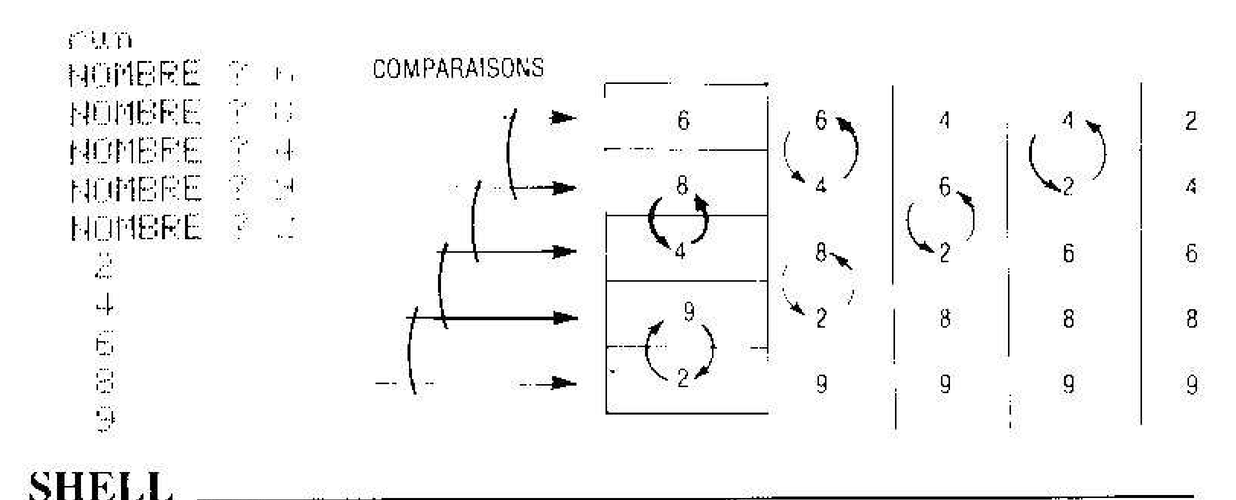
- s'il n'y a pas eu d'inversion, c'est que tous les éléments sont dans l'ordre ;
- s'il y a eu au moins une inversion, nous nous repositionnons en début de table et nous recommençons.

Après la première exploration de la table, le plus grand des éléments de la table est en fin de table.
Après la deuxième exploration de la table, le plus grand des éléments restants est en avant-dernière position.
etc.
Ainsi, après plusieurs passages dans la table (N pour éléments) tous les éléments seront dans l'ordre.
Sur l'exemple, nous explorons systématiquement toute la table. En fait, nous pourrions réduire la taille de la table explorée de 1 à chaque passage.
La variable "INV" sert de témoin pour tester en fin de table si on a, au cours de l'exploration de la table, fait une inversion.

Remarque : la méthode de tri utilisée ("RIPPLE SORT") qui a l'avantage d'être simple à comprendre (et à programmer) est rarement utilisée en pratique dès que le nombre d'éléments à trier devient important. C'est en effet une des plus lentes que l'on puisse trouver.

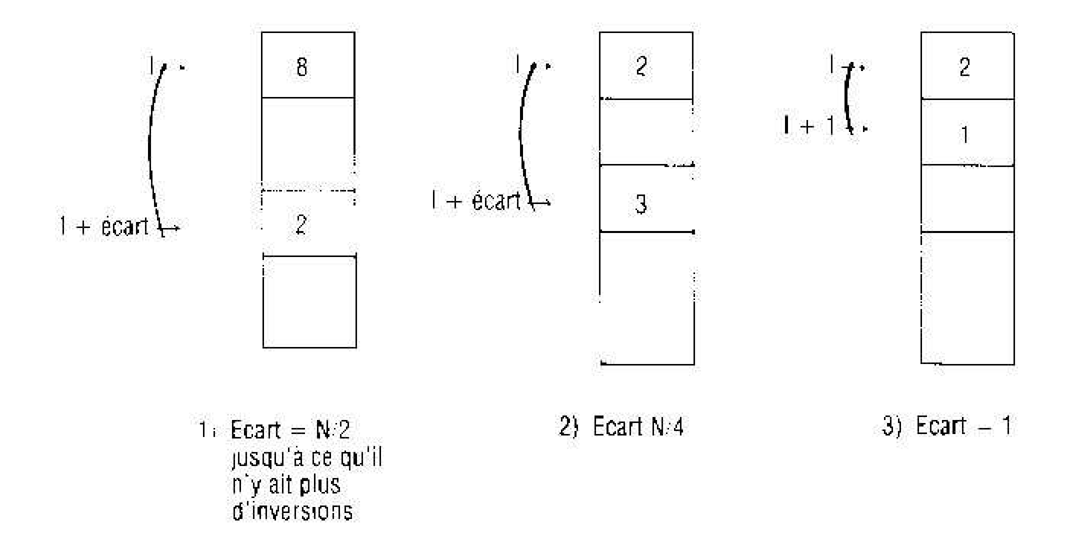
```
10 '   TRI DES NOMBRES D'UNE TABLE
20 '
30 '--------------------- ACQUISITION NOMBRES
40 N=5                       ' nombre d'elements
50 FOR I=1 TO N
60  INPUT "NOMBRE ",A(I)
70 NEXT I
80 '--------------------- TRI
90 INV=0
100 '
110 FOR I=1 TO N-1
120  IF A(I+1)<A(I) THEN SWAP A(I),A(I+1) :INV=1
130 NEXT I
140 IF INV=1 THEN 90      ' y a t-il eu inversion?
150 '----------------------- EDITION
160 FOR I=1 TO N
170  PRINT A(I)
180 NEXT I
```

→



SHELL

Avec les méthodes de tri du type RIPPLE, un grand nombre placé en début de table ne remonte que progressivement en fin de table. Avec SHELL, la comparaison s'effectue entre deux éléments séparés par un écart égal, au départ, à la moitié de la taille de la table.
Ainsi, un élément grand en début de table "remonte" plus vite en fin de table.



La comparaison se fait ensuite avec un écart égal à N/4 et s'achève avec un écart égal à 1 (comme RIPPLE). La méthode de SHELL/METZNER est deux fois plus rapide que SHELL.

```
10 '   TRI SHELL
20 '
30 NFICH=30
40 DIM A(NFICH)
50 FOR I=1 TO NFICH:A(I)=RND(1):NEXT I
60 TIME=0
70 '------------------------------
```

```
80 ECART=NFICH
90 '
100 ECART=INT(ECART/2):IF ECART<1 THEN 200
110 '
120 INV=0
130 FOR I=1 TO NFICH-ECART
140  L=I+ECART
150  IF A(L)<A(I) THEN SWAP A(I),A(L):INV=1
160  NEXT I
170 IF INV=1 THEN 120
180 GOTO 100
190 '------ EDITION
200 PRINT "TEMPS ";TIME/50
210 FOR I=1 TO NFICH:PRINT A(I):NEXT I
```

```
10 '    TRI SHELL-METZNER
20 '
30 NFICH=30
40 DIM A(NFICH)
50 FOR I=1 TO NFICH:A(I)=RND(1):NEXT I
60 TIME=0
70 '-------------------
80 ECART=NFICH
90 '
100 ECART=INT(ECART/2):IF ECART<1 THEN 230
110 J=1:K=NFICH-ECART
120 '
130 I=J
140 '
150 L=I+ECART
160 IF A(I)<=A(L) THEN 210
170 SWAP A(I),A(L)
180 I=I-ECART:IF I<1 THEN 210
190 GOTO 150
200 '
210 J=J+1:IF J>K THEN 100 ELSE 130
220 '---
230 PRINT TIME/50
240 FOR I=1 TO NFICH:PRINT A(I):NEXT I
```

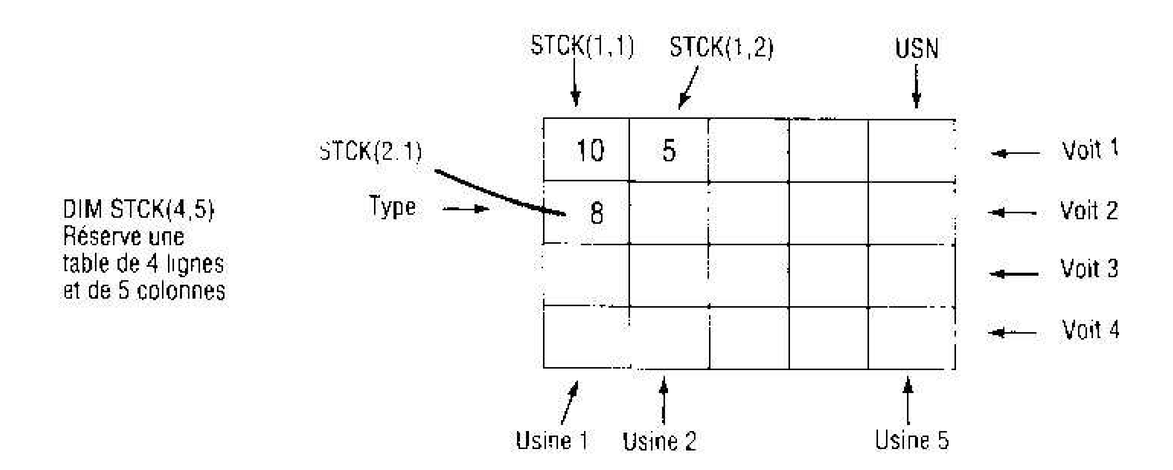
TABLES A PLUSIEURS DIMENSIONS

Plus généralement, les tables peuvent avoir plusieurs dimensions déclarées par :

DIM nom-table (dim1, dim2,...)

Soit une table à 2 dimensions contenant des stocks de voitures. de différents types et de différentes usines.

DIM STCK(4,5)
Réserve une
table de 4 lignes
et de 5 colonnes

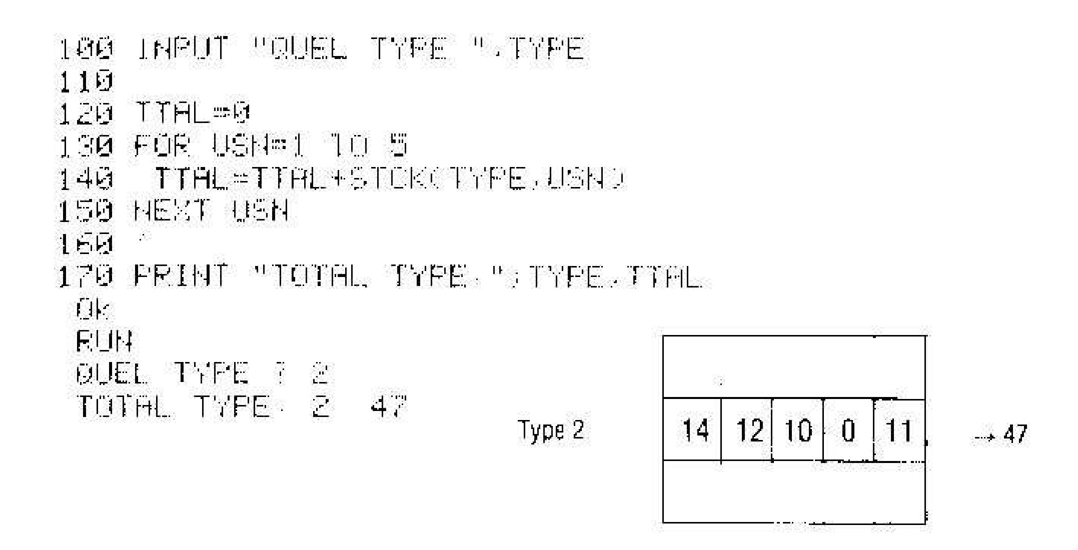


Documentons la table à l'aide de l'instruction INPUT :

```
10 DIM STCK(4,5)
20 '
30 FOR TYPE=1 TO 4
40  FOR USN=1 TO 5
50   PRINT "STOCK/TYPE:";TYPE;"USINE:";USN;
60   INPUT STCK(TYPE,USN)
70  NEXT USN
80 NEXT TYPE
 Ok
 RUN
 STOCK/TYPE: 1 USINE: 1 ? 10
 STOCK/TYPE: 1 USINE: 2 ? 5
 STOCK/TYPE: 1 USINE: 3 ? 3
 STOCK/TYPE: 1 USINE: 4 ? 7
 STOCK/TYPE: 1 USINE: 5 ? 9
 STOCK/TYPE: 2 USINE: 1 ? 8
 STOCK/TYPE: 2 USINE: 2 ?
```

Lorsque la table est documentée, pour connaître le nombre de véhicules d'un type, toutes usines confondues, on fait :

```
100 INPUT "QUEL TYPE ";TYPE
110
120 TTAL=0
130 FOR USN=1 TO 5
140  TTAL=TTAL+STCK(TYPE,USN)
150 NEXT USN
160 '
170 PRINT "TOTAL TYPE:";TYPE;TTAL
 Ok
 RUN
 QUEL TYPE ? 2
 TOTAL TYPE: 2  47
```



Comme pour les tables à 1 dimension, la déclaration d'une table à plusieurs dimensions, toutes inférieures ou égales à 10, n'est pas nécessaire.
Alors qu'une table et une variable peuvent avoir un même nom, plusieurs tables, même si elles ont des nombres de dimensions différents, ne peuvent avoir le même nom.

Exemple : `10 DIM A(7):DIM A(7,8)` est interdit

ERASE nom-table

Efface une table en mémoire centrale. La place est récupérée par BASIC et la table peut être à nouveau dimensionnée.

```
10 DIM A$(50)                    ' dimension 50
20 FOR I=1 TO 50
30  A$(I)=STR$(I)
40 NEXT I
50 '
60 PRINT FRE(0);FRE(X$)
70 ERASE A$                      ' effacement
80 PRINT FRE(0);FRE(X$)
90 '
100 DIM A$(15)                   ' dimension 15
run
 12044  59
 12205  200
```

ERREURS CLASSIQUES

Lorsqu'une table n'est pas dimensionnée par DIM, elle est dimensionnée par défaut avec 10 éléments par BASIC dès qu'un élément de la table est référencé en lecture ou en écriture.
Si une instruction DIM est exécutée après le dimensionnement par défaut, le message REDIMENSIONNED ARRAY est envoyé par BASIC.

```
10 A(4)=15  ' dimensionnement implicite
20 DIM A(20)
```

Il faut dimensionner la table avant de référencer un élément.
Lorsqu'une table n'est pas dimensionnée explicitement par DIM et que l'on essaie de référencer l'élément 11, on obtient le message SUBSCRIPT OUT OF RANGE.

```
10 FOR I=1 TO 15
20  PRINT A(I);
30 NEXT I
```

Il faut ajouter : 5 DIM A(15).

OCCUPATION MÉMOIRE

Par défaut, les tables numériques occupent 8 octets pour chaque élément (double précision)
Les tables déclarées entières (%) occupent 2 octets par élément.

```
10 PRINT FRE(0)
20 DIM A(100)
30 PRINT FRE(0)
run
 12395
 11579
```

```
10 PRINT FRE(0)
20 DIM A%(100)
30 PRINT FRE(0)

 12394
 12184
```

LES CHAÎNES DE CARACTÈRES

- LEFT$ ■ LEN ■ ASC ■ STRING$ ■ HEX$
- RIGHT$ ■ STR$ ■ CHR$ ■ SPACE$ ■ BIN$
- MID$ ■ VAL ■ INSTR ■ OCT$

L'affectation d'une valeur à une chaîne s'écrit :

nom de chaîne = « suite de caractères »

Exemple : 10 NOM$ = « DUPONT »

Les « » indiquent que DUPONT doit être interprété comme une chaîne de caractères et non comme une variable. La longueur d'une chaîne de caractères, qui n'a pas à être déclarée, peut varier en cours d'exécution du programme (jusqu'à 255). De même, la longueur de chaque élément d'une table de chaînes peut varier dynamiquement.
L'espace prévu pour les chaînes est de 200 caractères par défaut. **CLEAR "ESPACE CHAINE"** permet de définir cet espace. **PRINT FRE (X$)** donne l'espace libre pour les chaînes.
La concaténation (réunion) de chaînes de caractères est réalisée par l'opérateur noté « + ».

```
10 NOM$="DUPONT"
20 PREN$="JEAN"
30 NP$=NOM$+PREN$
40 PRINT NP$
OK
RUN
DUPONTJEAN
```

COMPARAISONS

La comparaison de chaînes de caractères se fait avec les opérateurs :

=, <, >, >=, <=, <>

Les chaînes sont comparées caractère par caractère de la gauche vers la droite jusqu'à ce que l'un des caractères d'une chaîne soit plus grand que l'autre (code ASCII supérieur). C'est alors cette chaîne qui est considérée comme la plus grande ("DURAND" est plus grand que "DUPONT").
Si tous les caractères sont égaux, les chaînes sont considérées comme égales.

```
10 INPUT "1ER NOM ";N1$
20 INPUT "2EME NOM ";N2$
30 IF N1$>N2$ THEN PRINT N1$;" PLUS GRAND QUE ";N2$
40 IF N1$<N2$ THEN PRINT N1$;" PLUS PETIT QUE ";N2$
50 GOTO 10
 RUN
 1ER NOM ? DURAND
 2EME NOM ? DUPOND
 DURAND PLUS GRAND QUE DUPOND
```

Le programme ci-dessous teste une réponse au clavier.

```
10 R$="":INPUT "REPONSE (OUI/NON) ";R$
20 IF R$="OUI" THEN PRINT "VOUS AVEZ DIT OUI"
30 IF R$="NON" THEN PRINT "VOUS AVEZ DIT NON"
40 IF R$="" THEN PRINT "VOUS AVEZ APPUYE SUR <RETURN>"
50 GOTO 10
```

Remarquez l'initialisation de R$ en 10 (chaîne vide) ainsi que le test d'une chaîne vide en 40 (IF R$ = " ").

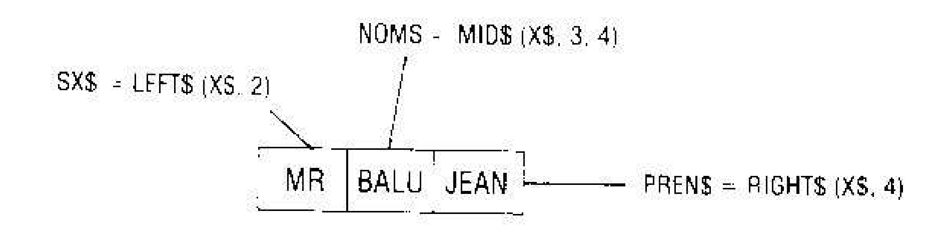
LEFT$, RIGHT$, MID$

LEFT$, RIGHT$, MID$ permettent d'accéder respectivement aux caractères de gauche, de droite et de l'intérieur d'une chaîne.

LEFT$ (CHAINE, longueur à prendre à gauche)
RIGHT$ (CHAINE, longueur à prendre à droite)
MID$ (CHAINE, position début, longueur à prendre)

"CHAINE" "longueur à prendre" et "position début" peuvent être des expressions. Les valeurs de "longueur à prendre" et de "position début" doivent être comprises entre 0 et 255.

Exemple :




```
10 X$="MRBALUJEAN"
20 SX$=LEFT$(X$,2)      ' 2 caracteres a gauche
30 PREN$=RIGHT$(X$,4)   ' 4 carcateres a droite
40 NOM$=MID$(X$,3,4)    ' 4 caracteres a partir du 3eme
50 PRINT SX$;" ";PREN$;" ";NOM$
 Ok
 run
 MR JEAN BALU
```

Dans l'exemple ci-dessous, nous testons la première lettre de la réponse :

```
10 INPUT "REPONSE (OUI/NON) ";R$
20 IF LEFT$(R$,1)="O" THEN GOTO 100
```

Autre exemple :

```
10 NOM$="DUBONET"
20 FOR I=1 TO 7
30  PRINT LEFT$(NOM$,I)
40 NEXT I
Ok
run
D
DU
DUB
DUBO
DUBON
DUBONE
DUBONET
```

Si "**longueur à prendre**" spécifiée est plus grande que la longueur de la chaîne, le résultat est la chaîne elle-même.
Lorsque "**longueur à prendre**" n'est pas précisée dans MID$, cette fonction devient équivalente à RIGHT$.
Si "**position début**" spécifiée dans MID$ est plus grande que la chaîne elle-même, une chaîne vide est retournée.

MID$(CHAINE1, position début, longueur à remplacer) = CHAINE2

Permet, à partir de "**position début**" dans "CHAINE1" et sur la longueur spécifiée, de remplacer des caractères par ceux de "CHAINE2".

```
10 X$="MRBALUJEAN"
20 MID$(X$,3,4)="XXXX"
30 PRINT X$
Ok
run
MRXXXXJEAN
```

Attention : ne permet pas l'insertion ou la suppression de caractères mais seulement la substitution.
Si "CHAINE2" est plus longue que "longueur à remplacer", seuls les premiers caractères de "CHAINE2" sont pris en considération.
Si "CHAINE2" est plus courte que "longueur à remplacer", il n'y a substitution des caractères que sur une longueur égale à celle de "CHAINE2".

LEN (chaîne)

Donne la longueur d'une expression chaîne.

```
10 NOM$="DUPONT"
20 L=LEN(NOM$)
30 PRINT L
Ok
run
 6
```

STR$(X)

Convertit une expression numérique X en une chaîne de caractères.

```
10 X=123
20 X$=STR$(X)
30 PRINT X$,LEN(X$)
Ok
run
 123           4
```

Remarque : Le premier caractère de la chaîne est réservé pour le signe ":" C'est un espace pour un nombre positif et un signe "–" pour un nombre négatif.

```
10 PRINT MID$(STR$(123),2,3) --> 1
```

VAL (chaîne)

Fonction inverse de STR$, elle donne la valeur numérique d'une expression chaîne.

```
10 X$="123 FRANCS"
20 X=VAL(X$)
30 PRINT X
Ok
run
 123
```

Si le premier caractere n'est pas un caractère décimal, un espace, un signe "+", un signe "–" ou ":" le résultat est égal à zéro.

```
10 PRINT VAL(" 123")
20 PRINT VAL("+123")
30 PRINT VAL("A122")
Ok
RUN
 123
 123
 0
```

ASC (caractère)

Chaque caractère a un code interne (code ASCII) auquel on accède par la fonction ASC (CARACTÈRE)

```
10 X$="A"
20 X=ASC(X$)
30 PRINT X
Ok
RUN
 65
```

ASC (chaîne)

Donne le code ASCII du premier caractère d'une expression chaine.

```
10 PRINT ASC("BONJOUR")  -->66
```

Une chaîne nulle comme argument provoque une erreur (Illegal Fonction Call).

```
10 X$="":PRINT ASC(X$)
```

CHR$(X)

Fonction inverse de la fonction ASC, elle permet de générer des caractères ayant pour code ASCII la valeur de X. Cette valeur doit être comprise entre 0 et 255. X peut être une constante, une variable ou une expression.

```
10 FOR C=65 TO 65+25
20  PRINT CHR$(C);
30 NEXT C
Ok
RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

CONVERSION MAJUSCULES → MINUSCULES :

En ajoutant 32 aux codes ASCII des caractères majuscules, on obtient des caractères minuscules.

Exemples :

```
10 PRINT CHR$(ASC("A")+32)  --> a
```

```
10 INPUT "VOTRE NOM (minuscules) ";NOM$
20 FOR P=1 TO LEN(NOM$)
30  PRINT CHR$(ASC(MID$(NOM$,P,1))-32)
40 NEXT P
 Ok
 run
 VOTRE NOM (minuscules) ? roulet
 ROULET
```

CHR$(X) est utilisé pour envoyer des "caractères de contrôle" aux périphériques (écran, imprimante).

CHR$(8)	provoque un retour arrière du curseur.
CHR$(10)	provoque un passage à la ligne (sans retour en début de ligne).
CHR$(13)	provoque un retour en début de ligne.
CHR$(200)	affiche un pavé.

Exemples divers :

■ Suppression d'un caractère à droite d'une chaîne :

```
10 NOM$="DUPONT"
20 X$=LEFT$(NOM$,LEN(NOM$)-1)
30 PRINT X$
 Ok
 run
 DUPON
```

■ Normalisation d'une chaîne à une longueur fixe :

```
5  complete par des espaces a droite
10 NOM$="DUPONT"
20 Y$=LEFT$(NOM$+"        ",8)
30 PRINT Y$;"XXX"
Ok
run
DUPONT  XXX
```

■ Insertion d'un caractère dans une chaîne :

```
10 Z$="AAAAA"
20 X$="B":P=3    ' caractere et position
30 Z$=LEFT$(Z$,P)+X$+RIGHT$(Z$,LEN(Z$)-P)
40 PRINT Z$
 Ok
 run
 AAABAA
```

■ Remplissage par des zéros à gauche :

```
10 X=123
20 X$=RIGHT$(STR$(100000!+X),5)
30 PRINT X$
Ok
run
00123
```

INSTR (position départ, chaîne, chaîne cherchée)

Recherche la position d'une chaîne dans une autre. Par défaut, la position de départ est égale à 1 :

- Si la chaîne cherchée n'est pas trouvée, le résultat est égal à 0.
- Si la chaîne cherchée est nulle, le résultat est la position de départ spécifiée.
- Si la position de départ est supérieure à la longueur de la chaîne où s'effectue

la recherche, le résultat est nul.

```
10 X$="DUPONT.JEAN"
15 '
20 P=INSTR(X$,".")      ' recherche caractere '.'
30 NOM$=LEFT$(X$,P-1)
40 PRINT NOM$,P
 Ok
 run
 DUPONT          7
```

Le programme ci-dessous vérifie si un nom appartient à un ensemble.

```
10 E$="JEAN PIERRE PAUL JACQUES"
20 INPUT "Nom ";NOM$
30 P=INSTR(E$,NOM$)
40 IF P=0 THEN PRINT "Erreur":GOTO 20
50 GOTO 20
```

Cette séquence d'instructions permet de répondre à une question "Mode ?" non pas par un chiffre, mais par une lettre (plus mnémonique).

```
10 INPUT "Mode (C,A,P,R,M,N) ";M$       ' entrer C,A,P....
15 '
20 P=INSTR(" CAPRMN",M$)                ' Position caractere frappe?
25 IF P<2 THEN BEEP:GOTO 10             ' chaine vide?
30 '
40 ON P-1 GOTO 100,200,300,400,500,600
50 STOP
60 '
100 PRINT "Ligne 100":STOP
200 PRINT "Ligne 200":STOP
```

Remarque : Si M$ est une chaîne vide, P est égal à 1. C'est ce qui explique la présence d'un espace dans l'instruction 20 devant "CAPRMN"
La séquence d'instructions ci-dessous vérifie si un caractère frappé appartient bien à un ensemble de caractères autorisés : (chiffres plus le "." sur l'exemple).

```
10 C$=INPUT$(1)                     ' lecture 1 caractere
15 '
20 P=INSTR("0123456789.",C$)'       position caractere frappe
30 IF ASC(C$)=13 THEN 100           ' RETURN?
35 IF P=0 THEN BEEP:GOTO 10
40 LIG$=LIG$+C$                     ' ajout caractere frappe
50 PRINT C$;                        ' affichage
60 GOTO 10
90 '
100 PRINT:PRINT LIG$
```

STRINGS$ (nombre de fois, chaîne)

Génère une chaîne de caractères égale à la chaîne spécifiée, multipliée par le nombre de fois indiqué.

```
10 X$=STRING$(10,".")     ' chaine de 10 '.'
15 '
20 PRINT X$
 Ok
 run
 ..........
```

SPACE$(X)

Génère une chaîne de X espaces. X peut être une expression.

```
10 X$="DUPONT"+SPACE$(5)+"JEAN"
15 '
20 PRINT X$
Ok
run
DUPONT     JEAN
```

Notons que la fonction SPC ne génère pas une chaîne d'espaces et n'est utilisable qu'avec l'instruction PRINT.

OCT$ (expression)

Génère une chaîne qui représente la valeur octale de l'argument décimal .

```
10 PRINT OCT$(24)  -->30
```

HEX$ (expression)

Fournit une chaîne représentant la valeur hexadécimale de la valeur donnée. La valeur fournie doit être entière (−32768 —>32767).

```
10 X$=HEX$(14)
20 PRINT X$

RUN
E
```

BIN$ (expression)

Fournit une chaîne représentant la valeur binaire de la valeur donnée.

```
10 PRINT BIN$(127)

1111111
```

Remarque sur les chaînes : Lorsqu'une chaîne voit sa longueur changer, elle est déplacée dans l'espace prévu pour les chaînes de caractères. La place occupée par l'ancienne chaîne reste perdue jusqu'à ce que l'espace chaîne soit réorganisé par BASIC.
Pour éviter ces réorganisations (longues), on pourra utiliser **MID$ (X$,P,L)**-"**XX**" qui modifie la chaîne sans la déplacer.
Pour les tris, utiliser **SWAP** qui échange les descripteurs dans déplacer les chaînes.

Le programme ci-dessous remplit une table de chaîne.
La concaténation "**BASIC**" + **STR$(I)** utilise l'espace chaîne et provoque des réorganisations. En augmentant l'espace chaîne, le temps de remplissage est notablement réduit.

```
10 '------------------ temps reorganisation espace chaine
15 '
20 CLEAR 1700
30 N=200
40 DIM T$(N)
50 '
60 TIME=0
70 FOR I=1 TO N
80  T$(I)="BASIC"+STR$(I)
90 NEXT I
100 PRINT "Temps remplissage:",TIME/50
110 TIME=0
120 PRINT FRE(X$)
130 PRINT "Temps:",TIME/50

Temps remplissage  15.52
 8
Temps  3.94
```

```
20 CLEAR 3000

Temps remplissage: 1.68
 1308
Temps: 3.94
```

LES ÉDITIONS

- PRINT
- PRINT,
- PRINT;
- PRINT TAB
- PRINT SPC
- PRINT USING
- POS et CSRLIN
- WIDTH
- LPOS(0)

PRINT expression

Un simple PRINT d'une constante, variable ou expression affiche la valeur de celle-ci puis provoque un retour en début de ligne et un saut de ligne.

```
10 X=123
20 PRINT X
Ok
run
 123
```

PRINT,

L'impression de plusieurs valeurs sur une même ligne peut se faire simplement en séparant dans l'instruction les noms des variables ou les expressions par des virgules.

```
10 SOMME=200:NOMBRE=10
20 PRINT SOMME,NOMBRE,SOMME/NOMBRE
Ok
run
 200           10
 20
```

Mais dans ce cas, l'impression des valeurs est faite selon un format standard (colonnes 0 et 14). Les valeurs sont cadrées à gauche.
Sur l'exemple, la troisième valeur devrait être affichée en colonne 28. Mais le nombre d'espaces entre la colonne 28 et la colonne droite de l'écran étant inférieur à 14, la troisième valeur est affichée au début de la ligne suivante.

PRINT;

Un **point-virgule** en fin d'instruction PRINT empêche le saut de ligne : les chaînes de caractères sont concaténées, les valeurs numériques sont suivies par un espace et précédées soit par un espace pour les nombres positifs, soit par le signe "-" pour les nombres négatifs.

```
10 NOM$="DUPONT"
20 PREN$="JEAN"
30 PRINT NOM$;
40 PRINT PREN$
Ok
run
DUPONTJEAN
```

```
10 PRINT "DUPONT";"   ";"JEAN"   --> DUPONT   JEAN

10 PRINT 123;456,-789 --> 123456! -789
```

PRINT TAB(X)

La fonction TAB(X) permet de positionner directement le curseur à l'intérieur d'une ligne en colonne X, X peut être une expression. TAB(0) spécifie la première colonne.

```
10 A=123 B=456
20 PRINT A TAB(15) B
Ok
run
 123            456
```

"456" est affiché en colonne 15.

Attention : Le curseur ne peut revenir en arrière. Si X spécifié est plus petit que la position courante du curseur, TAB (X) est sans effet. (cf. LOCATE chapitre "L'écran").

PRINT SPC(X)

Avec la fonction SPC(X), X blancs sont imprimés à partir de la position courante :

```
10 NOM$="DUPONT":PR$="JEAN"
20 PRINT PR$;SPC(3);NOM$
Ok
run
JEAN   DUPONT
```

PRINT USING

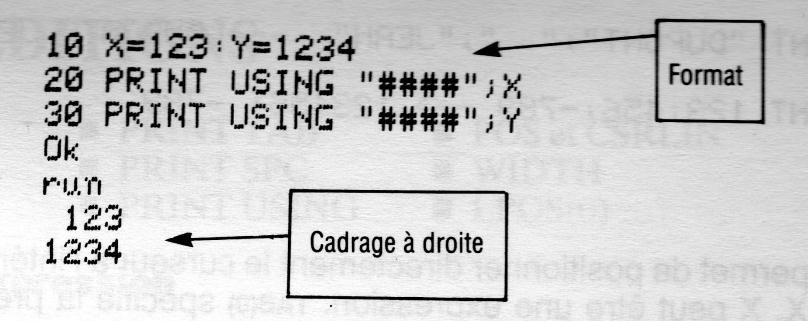
Considérons maintenant l'outil d'édition le plus puissant du BASIC : le "PRINT USING"

VARIABLES NUMÉRIQUES :

Sans le PRINT USING les valeurs numériques sont cadrées à gauche. Or, c'est généralement à droite qu'elles doivent être cadrées.

☐ **PRINT USING « # # # # # » ; expression numérique**

Un format défini par une chaîne de # nous permet de cadrer les nombres à droite. Chaque # représente la position d'un chiffre.



Sans le PRINT USING, nous aurions obtenu des chiffres cadrés à gauche.

```
123
1234
```

La partie entre guillemets qui représente le format peut aussi être définie par une variable chaîne de caractères :

```
10 X=123:Y=1234
20 FMT$="####"
30 PRINT USING FMT$;X
40 PRINT USING FMT$;Y
Ok
run
 123
1234
```

□ **PRINT USING** « # # # #.# # » **; expression numérique :**

Le nombre de chiffres après la virgule qui doivent être imprimés est précisé dans le format par le nombre de # après le "."

```
10 X=123.456
20 PRINT USING "####.##";X
Ok
run
 123.46 )
```

2 chiffres après la virgule

On remarque que l'arrondi est assuré automatiquement.

□ **PRINT USING «LIBELLÉ »** « # # # #.# # »**, valariable numérique**

Un libellé peut être inséré dans le format.

```
10 SOMME=123.456
20 PRINT USING "Total: ####.## FRANCS";SOMME
Ok
run
Total:   123.46 FRANCS
```

FORMATS MULTIPLES :

Plusieurs formats peuvent être spécifiés dans une seule instruction.

```
10 SOMME=1234.567#:TVA=13
20 PRINT USING "Total: ####.## TVA ### ";SOMME,TVA
Ok
run
Total: 1234.57 TVA  13
```

Si le format est le même pour plusieurs variables, on fait :

```
10 X=123:Y=456
20 PRINT USING "####.##";X,Y
Ok
run
 123.00 456.00
```

(+) Le signe "+" n'est imprimé que s'il est prévu dans le programme.

```
10 PRINT USING " ####";123   --> 123
20 PRINT USING "+####";123   -->+123
30 PRINT USING "+####";-123  -->-123
Ok
```

(−) Un signe "−" en fin de format provoque l'impression du signe " " à la fin d'un nombre négatif.

```
PRINT USING "####.#-";-15.1  -->15.1-
PRINT USING "####.#-";15.1   -->15.1
```

(**) 2 "*" placés en tête de format provoquent le remplissage par des "*" des positions inoccupées à gauche du nombre imprimé. En outre ces 2 "*" spécifient des positions pour 2 chiffres supplémentaires.

```
10 PRINT USING "**####";123   -->**123
20 PRINT USING "**####";1234  -->*1234
```

($$) 2 "$" en tête de format provoquent l'impression du caractère "$" à gauche du nombre imprimé.

```
10 PRINT USING "$$####";123   --> $123
```

(**$) "**$" combine les effets "**" et "$$".

```
10 PRINT USING "**$####";123 -->***$123
```



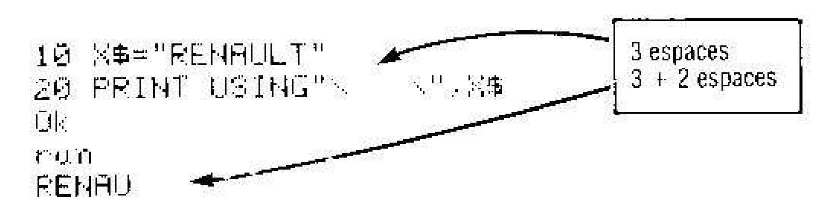
Si le nombre de positions spécifié dans le format est insuffisant pour la valeur à imprimer, le message % est imprimé devant la valeur.

```
10 PRINT USING "###",12345 --> %12345
```

CHAÎNES DE CARACTÈRES

□ PRINT USING «\   \» ; expression chaîne :

Le nombre d'espaces entre les "\" définit le nombre de caractères à imprimer−2.



□ PRINT USING « ! » ; expression chaîne :

Imprime le premier caractère d'une expression chaîne.

□ PRINT « & » ; expression chaine :

Spécifie une longueur variable de chaîne. La chaine imprimée est égale à la chaîne spécifiée.

```
10 NOM$="ROULET":PREN$="NICOLAS"
20 PRINT USING "&   !";NOM$,PREN$
Ok
RUN
ROULET   N
```

Programme pour tester PRINT USING

Le programme suivant permet d'entrer par INPUT à la fois le format et le nombre à imprimer.

```
10 INPUT "Format,Nombre ";FMT$,NOMBRE

20 PRINT USING FMT$;NOMBRE
30 GOTO 10
Ok
run
Format,Nombre ? ####.##,123.456
  123.46
```

POS(0)-CSRLIN

Donnent les coordonnées du curseur.

```
10 LOCATE 10,8
20 PRINT "NICOLAS";
30 PRINT POS(0);CSRLIN
Ok
run
                   NICOLAS 17  8
```

WIDTH largeur

Détermine la largeur d'édition à l'écran. L'affichage est centré.

```
10 WIDTH 20
```

Par défaut, la largeur est 37 en "SCREEN0" et 28 en "SCREEN1".

EDITIONS IMPRIMANTE :

Pour afficher des résultats sur imprimante utiliser LPRINT au lieu de PRINT.
Pour éditer sur imprimante, on peut également ouvrir un canal vers l'imprimante.

```
10 OPEN "LPT:" FOR OUTPUT AS #1
20 PRINT #1,"COUCOU"
30 CLOSE#1
```

LPOS(0)

Fournit la position de la tête d'impression de l'imprimante.

ÉDITIONS IMPRIMANTE/ÉCRAN :

Un même programme peut aiguiller des résultats vers l'écran ou l'imprimante.

```
10 INPUT "ECRAN OU IMPRIMANTE (E/I) ";M$
15 '
20 IF M$="E" THEN F$="CRT:"
30 IF M$="I" THEN F$="LPT:"
40 '
50 OPEN F$ FOR OUTPUT AS #1
60 '
70 PRINT#1,"BLABLA"
80 CLOSE #1
90 GOTO 10
```

Si l'édition se fait par l'intermédiaire de "CRT:","PRINT," sépare les valeurs par 14 espaces.

```
'
5 '---------- CRT:
10 OPEN "CRT:" FOR OUTPUT AS #1
20 '
30 PRINT #1,1236,73
```

Le programme ci-dessous affiche sur imprimante la liste des codes ASCII entre 32 et 128.

```
10 ' Edition des codes ASCII
20 '
30 FOR C=32 TO 63
40  LPRINT C;CHR$(C),
50  LPRINT C+32;CHR$(C+32),
60  LPRINT C+64;CHR$(C+64)
70 NEXT C
```

```
 32                 64 @              96 `
 33 !               65 A              97 a
 34 "               66 B              98 b
 35 #               67 C              99 c
 36 $               68 D              100 d
 37 %               69 E              101 e
 38 &               70 F              102 f
 39 '               71 G              103 g
 40 (               72 H              104 h
 41 )               73 I              105 i
 42 *               74 J              106 j
```

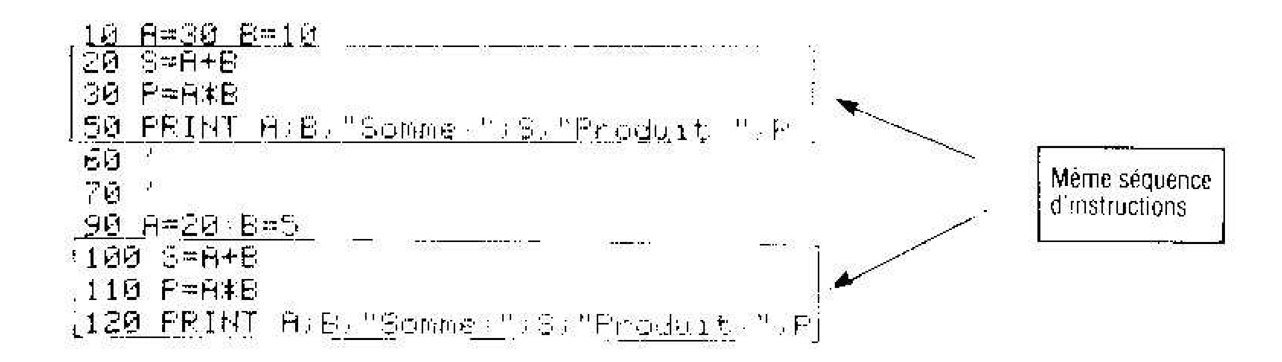
DÉCOUPAGE DES PROGRAMMES | 4

LES SOUS-PROGRAMMES

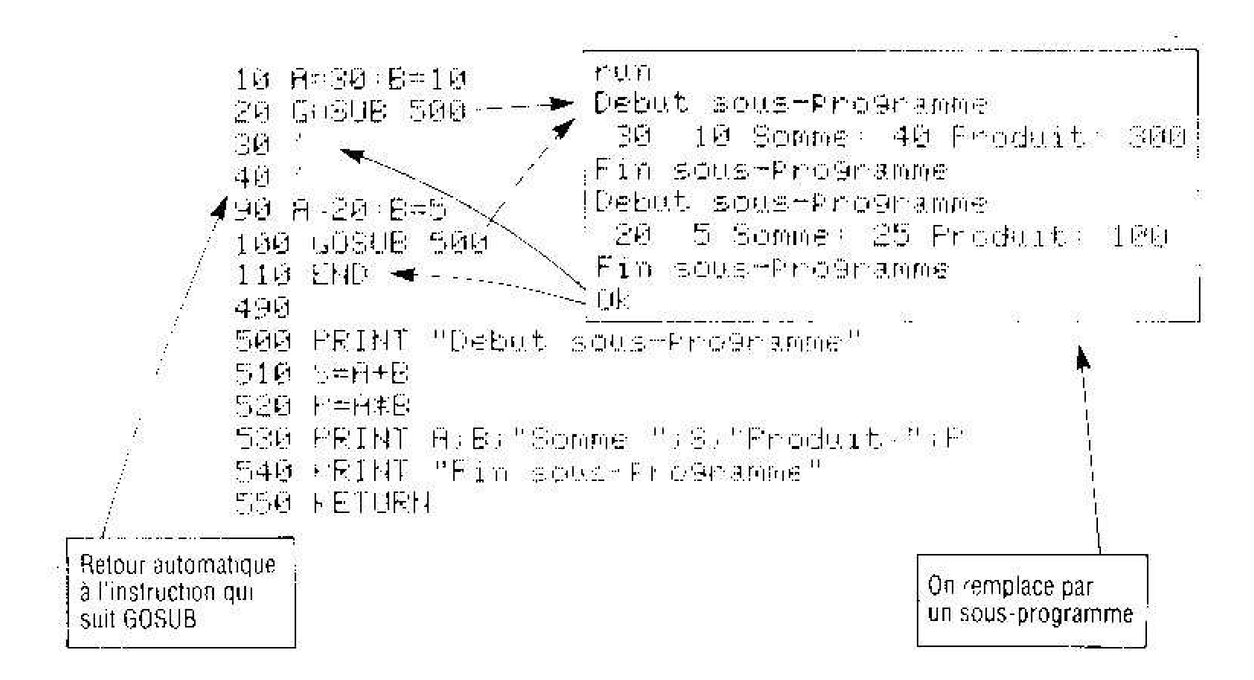
■ GOSUB... RETURN

GOSUB RETURN

Il est fréquent qu'une même séquence d'instructions soit utilisee plusieurs fois dans un programme. Un sous-programme permet d'écrire une seule fois cette séquence qu'il suffit d'appeler de différents endroits du programme par **GOSUB N° D'INSTRUCTION**.

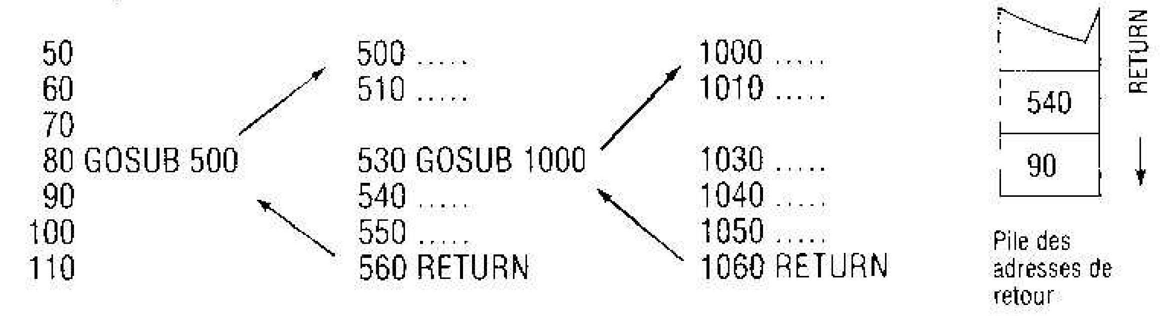


20 GOSUB 500 provoque un branchement du programme en 500 (comme le ferait GOTO 500) mais l'instruction RETURN (RETOUR)) placée à la fin du sous-programme provoque un **retour automatique** après l'instruction qui suit GOSUB 500, c'est-à-dire l'instruction 30 sur l'exemple.
Pour le 2e appel du sous-programme en 100, le retour se fait en 110.



Un sous-programme peut lui-même en appeler un autre.

Les adresses de retour (90 et 540 sur l'exemple) sont gérées par Basic à l'aide d'une pile.



Les instructions sont exécutées dans l'ordre suivant :

50 60 70 80	
500 510 520 530	Début du 1er sous-programme
1000 1010 ... 1050 1060	2e sous-programme
540 550 560	Fin du 1er sous-programme
90 100	Retour au programme principal

N'essayez pas de sortir d'un sous-programme par GOTO, ni d'entrer dans un sous-programme par GOTO. Les sous-programmes sont utilisés non pas seulement pour économiser de la place mémoire, mais aussi (et plutôt) pour les raisons suivantes :

- Par souci de clarté : le "programme principal" est plus court.
- Les sous-programmes permettent éventuellement de répartir la programmation entre plusieurs programmeurs.
- La mise au point se fait sous-programme par sous-programme. Ainsi on progresse plus sûrement dans la mise au point de l'ensemble du programme.
- Dans le cas où la même séquence d'instructions est répétée, une modification dans cette séquence doit être faite aux différents endroits où elle a été écrite (fastidieux).

- Lorsqu'un sous-programme est modifié, la "zone d'intervention" est bien délimitée ; tout le programme ne risque pas d'être remis en cause.
- Les sous-programmes ont aussi l'avantage, lorsqu'en cours de mise au point, il faut "restructurer" le programme, de permettre une certaine souplesse ; l'ordre d'appel des sous-programmes est à modifier mais ceux-ci ne changent pas.
- Plutôt que d'insérer une nouvelle séquence d'instructions correspondant à l'ajout d'une nouvelle fonction, on peut écrire un sous-programme qui est appelé quand nécessaire.

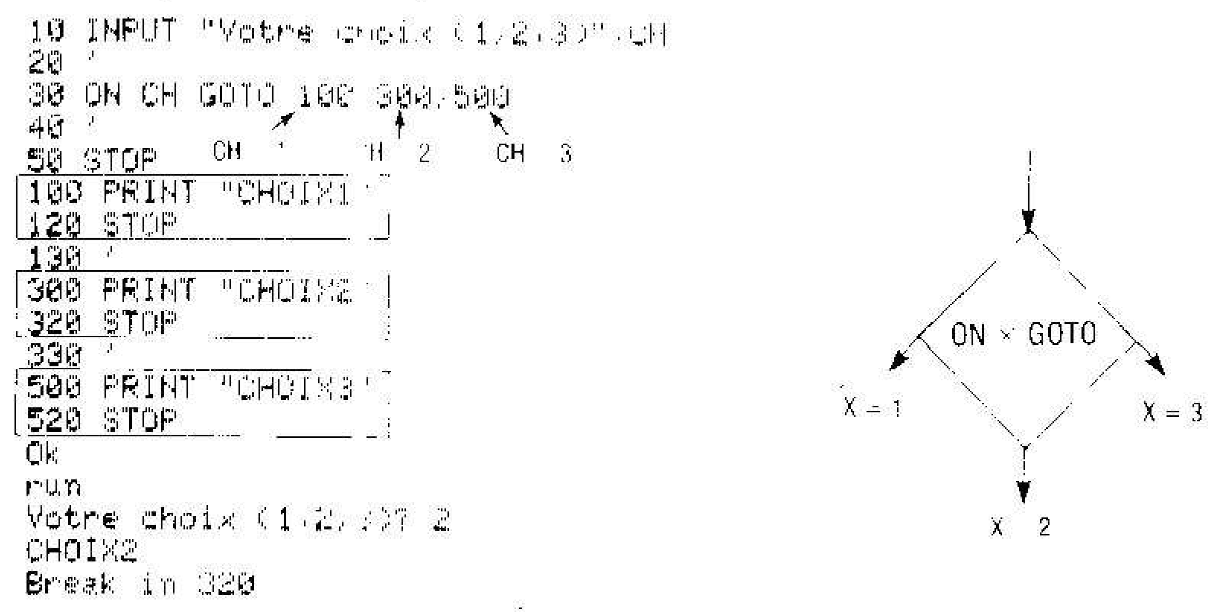
Les avantages des sous-programmes sont donc multiples. Aussi n'hésitera-t-on pas à en faire un usage abondant, même lorsqu'ils ne comportent que quelques instructions.

LES BRANCHEMENTS MULTIDIRECTIONS

- ON X GOTO...
- ON X GOSUB...

ON X GOTO n° ligne1, n° ligne2

Suivant la valeur d'une variable (ou d'une expression) : 1, 2, 3,... il y a branchement au n° de ligne1, n° ligne2, n° ligne3,...
S'il ne correspond pas de numéro de ligne à la valeur de la variable, c'est l'instruction qui suit "ON GOTO" qui est exécutée.




```
10 INPUT "Votre choix (1,2,3)";CH
20 '
30 ON CH GOTO 100,300,500
40 '
50 STOP
100 PRINT "CHOIX1"
120 STOP
130 '
300 PRINT "CHOIX2"
320 STOP
330 '
500 PRINT "CHOIX3"
520 STOP
Ok
run
Votre choix (1,2,3)? 2
CHOIX2
Break in 320
```

ON X GOSUB n° ligne1, n° ligne2,...

Fonctionne comme ON X GOTO, mais dès qu'une instruction RETURN est rencontrée, il y a retour automatique après l'instruction ON X GOSUB...

```
10 INPUT "Votre choix (1,2,3)";CH
20 '
30 ON CH GOSUB 100,300,500
40 PRINT "Suite"
50 GOTO 10
100 PRINT "CHOIX1"
120 RETURN
130
300 PRINT "CHOIX2"
320 RETURN
330
500 PRINT "CHOIX3"
520 RETURN
Ok
run
Votre choix (1,2,3)? 2
CHOIX2
Suite
Votre choix (1,2,3)?
```

LES FONCTIONS | 5

LES FONCTIONS ARITHMÉTIQUES

L'argument X peut être une constante, une variable ou une expression numérique.

ABS(X) Fournit la valeur absolue de X :
100 PRINT ABS(−35) → 35

ATN(X) Donne en radians l'arctangeante de X.

CDBL(X) Convertit X en un nombre double précision (avec 16 digits). Elle permet aussi d'effectuer une opération en double précision :
100 FOR I% = 1 TO 10 : PRINT 1/CDBL(I%) : NEXT I%

CINT(X) Convertit X en un entier avec arrondi :
100 PRINT CINT(1.6) → 2
110 PRINT CINT(−1.2) → −1

COS(X) Donne le cosinus de X exprimé en radians.

CSNG(X) Convertit X en un nombre simple précision (6 digits) :
100X = 1234.56789123 : A = CSNG(X) : PRINT A → 1234.57

EXP(X) Donne l'exponentielle de X.

FIX(X) Supprime les chiffres après la virgule :
100 PRINT FIX(2.2) → 2
110 PRINT FIX(−2.2) → −2

FRE(0) Donne la place libre en mémoire centrale (l'argument 0 n'est pas utilisé)

INT(X) Donne la partie entière de X avec arrondi :
100 PRINT INT(2.2) → 2
110 PRINT INT(−2.2) → −3

LOG(X) Fournit le logarithme (naturel) de X

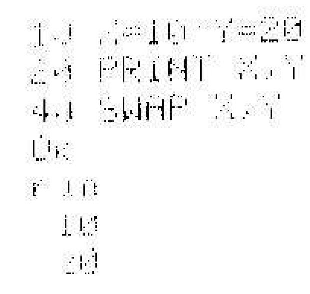
SGN(X) Donne le signe X : on obtient respectivement −1, 0, +1 pour les valeurs négatives, nulles, positives.

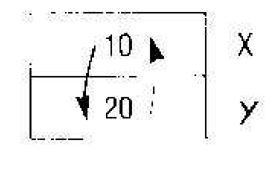
SIN(X) Donne le sinus de X exprimé en radians.

SQR(X) Donne la racine carrée de X. (X doit être positif)

TAN(X) Donne la tangente de X.

SWAP élément1, élément2 Les valeurs de "élément1" et de "élément2" sont échangées. Ces éléments doivent être des variables ou les éléments de tableaux du même type.





Pour les variables chaînes, SWAP échange les descripteurs de chaînes seulement, évitant ainsi les réorganisations de l'espace chaînes.

LA DÉFINITION DE FONCTIONS

En plus des fonctions internes (telles que SQR, SGN, INT, etc.), l'utilisateur peut définir ses propres fonctions par :

DEF FNXX (X,Y,Z,...) = EXPRESSION (X,Y,Z,...)

où XX représente un nom choisi par l'utilisateur pour identifier sa fonction (les règles d'appellation sont les mêmes que pour les variables) et X,Y,Z,... les arguments de la fonction.
Plus tard, cette fonction sera appelée par le programme avec les valeurs réelles des paramètres.

Exemples :

■ FONCTION D'ARRONDI

```
10 DEF FNAR(X)=INT(X+.5)
20 A=123.6
30 B=345.2
40 PRINT FNAR(A),FNAR(B)
Ok
run
 124           345
```

■ FONCTION MAXIMUM DE 2 NOMBRES

```
10 DEF FNMX(X,Y)=-((X>Y)*X+(X<=Y)*Y)
20 A=25:B=20
30 PRINT FNMX(A,B)
```

■ FONCTION D'ARRONDI AVEC DÉCIMALES

```
10 DEF FNAR(X)=INT(X*100+.5)/100
20 Y=FNAR(12.456):PRINT Y
Ok
run
 12.46
```

Une fonction ne peut être écrite que sur une seule ligne de 255 caractères au plus. C'est de préférence en tête du programme que sont écrites les fonctions de façon à être interprétées avant qu'elles ne soient appelées.

NOMBRES ALÉATOIRES ET HORLOGE | 6

LES NOMBRES ALÉATOIRES

Les nombres aléatoires sont essentiellement utilisés pour les programmes de jeu ou d'éducation.

RND(X) pour X > 0

Fournit un nombre aléatoire compris entre 0 et 1 (1 exclus).

```
10 FOR N=1 TO 3
20  PRINT RND(1);
30 NEXT N
Ok
RUN
 .59521943994623   .10658628050158
 .76597651772823
Ok
RUN
 .59521943994623   .10658628050158
 .76597651772823
```

Lors d'un second "RUN" la série est identique

Pour obtenir des nombres entiers entre 0 et 9 par exemple, il faut :

- multiplier le nombre obtenu par 10
- prendre la partie entière du résultat, avec la fonction INT (X)

```
10 FOR N=1 TO 4
20  X=RND(1)
30  Y=X*10
40  Z=INT(Y)
50  PRINT X;Y;Z
60 NEXT N
Ok
RUN
 .59521943994623  5.9521943994623  5
```

→

```
.10658628050158  1.0658628050158  1

.76597651772823  7.6597651772823  7

.57756392935958  5.7756392935958  5
```

Pour obtenir un nombre entier entre 1 et 10, il suffit d'ajouter 1.

```
10 FOR N=1 TO 6
20   PRINT INT(RND(1)*10+1);
30 NEXT N
Ok
RUN
 6  2  8  6  8  2  0  0  0
```

RND (X) pour X < 0

X négatif initialise une série aléatoire qui dépend de la valeur de X. Cette série est toujours la même pour une valeur de X.

```
5 X=RND(-7)
10 FOR N=1 TO 6
20   PRINT INT(RND(1)*10+1);
30 NEXT N
Ok
RUN
 7  7  10  9  10  6
```

RND (0)

Fournit le dernier nombre aléatoire.

```
10 FOR N=1 TO 3
20   PRINT RND(1)
30 NEXT N
35 PRINT RND(0)
Ok
RUN
 .59521943994623
 .10658628050158
 .76597651772823
 .76597651772823
```

Pour obtenir une serie aléatoire différente à chaque "RUN",

```
10 X=RND(-TIME)          initialise une
 serie aleatoire
20 FOR N=1 TO 3
30  PRINT RND(1)
40 NEXT N
Ok
run
 .6413586816692
 .52278659582001
 .901227263056 4
Ok
run
 .4704086616692
 .7797865852001
 .6882714638564
Ok
```

L'HORLOGE

- TIME
- ON INTERVAL GOSUB
- INTERVAL ON
- INTERVAL OFF
- INTERVAL STOP

TIME

La variable TIME donne le temps écoulé en 1/50 seconde.

```
10 TIME=0
20 FOR TP=1 TO 2000:NEXT TP
30 '
40 PRINT "JE TRAVAILLE POUR VOUS DEPUIS ";TIME/50;"SECONDES"
```

La valeur de TIME est comprise entre 0 et 65535. Elle ne peut mesurer des intervalles de temps supérieurs à 65535/50=1310 secondes.

ON INTERVAL=temps GOSUB n° ligne :

Définit le numéro de ligne vers lequel il y aura périodiquement débranchement du programme. Le temps est spécifié en 1/50e de seconde et doit être inférieur à 65535.

INTERVAL ON :

Valide la déclaration faite par ON INTERVAL GOSUB.

INTERVAL OFF :

Annule INTERVAL ON.

INTERVAL STOP :

Désactive la validation mais l'interruption est mémorisée jusqu'à ce que INTERVAL ON soit exécuté.

Le programme suivant affiche l'heure toutes les 4 secondes.

```
10 CLS
20 ON INTERVAL=200 GOSUB 120      ' toutes les 4 secondes
30 '
40 INTERVAL ON                    ' validation INTERVAL
50 '
60 LOCATE 1,1
70 B=B+1
80 PRINT "BOUCLE:",B
90 GOTO 60
100 END
110 '------------ sous-programme INTERVAL
120 T=T+1
130 LOCATE 10,10
140 PRINT T*4;"SECONDES"
150 RETURN
```

Ci-dessous, l'opérateur doit répondre dans un délai de 12 secondes.

```
10 CLS
20 ON INTERVAL=50 GOSUB 140      ' toutes les secondes
30 '
40 INTERVAL ON                   ' validation INTERVAL
50 '
60 LOCATE 1,1
70 PRINT "RePondez(O/N):"
80 '
90 R$=INKEY$:IF R$<>"" THEN 120
100 IF T>12 THEN PRINT "TROP TARD":END
110 GOTO 90
120 END
130 '------------ sous-Programme INTERVAL
140 T=T+1
150 LOCATE 10,10
160 PRINT T;"SECONDES"
170 RETURN
```

L'ACCÈS À LA MÉMOIRE ET ENTRÉES-SORTIES DIRECTES 7

L'ACCÈS À LA MÉMOIRE

- PEEK
- POKE
- VARPTR
- BASE
- VPEEK
- VPOKE
- VDP

PEEK (adresse mémoire)

Fournit en décimal le contenu d'un octet de la mémoire.

```
10 FOR M=1 TO 5
20  PRINT M,PEEK(M)
30 NEXT M
run
 1              195
 2              215
 3              2
 4              191
 5              27
```

1 [195]
↓ PEEK
195

POKE, adresse mémoire, valeur

Range une valeur exprimée en décimal (0→255) à l'adresse spécifiée.

```
Poke 50000,65        ' 65 dans 50000
Print Peek(50000)    ' affiche le contenu de 50000
```

65
↓
50000 [65]

Naturellement, il faut que la mémoire adressée soit modifiable.

Le programme ci-dessous affiche le contenu de la mémoire. Les caractères dont les codes sont superieurs à 31 sont affichés directement.

```
10 INPUT "ADRESSE DEBUT ";AM
20 INPUT "COMBIEN D'OCTETS ";N
30 '
40 FOR I=AM TO AM+N
45  IF I-12*INT(I/12)=0 THEN PRINT:PRINT I;TAB(7);":";
50  X=PEEK(I)
60  IF X=>32 THEN PRINT CHR$(X); ELSE PRINT USING "####";X;
80 NEXT I

 49140 :
 49152 :    0  27P  10    0 "ADRE
 49164 :SSE DEBUT ";
 49176 :AM    07P  20    0 "CO
 49188 :MBIEN D'OCTE
 49200 :TS ";N   07P  30    0:
 49212 :    0QP(    0 IA
 49224 :M y AMN    0yP-
 49236 :    0  I  15  12(
 49248 :I  15  12)  17 z :
 49260 :  I;(  24);":";
 49272 :    0P2   0 X(I
 49284 :)    0$P(    0  X
 49296 :   15   z  (X)
 49308 :;  :d    "##
 49320 :##";X;    00PP    0
 49332 :  I    0    0    0   8AME


 32460 :R^    1    9    0    9"bY
 32472 :MSX  system    0
 32484 :version 1.0   13
 32496 :   10    0MSX BASIC
 32508 :     0Copyright 1
 32520 :983 by Micro
 32532 :soft   13   10    0 Byte
 32544 :s free   0sê^  24    3
 32556 :sêszs
```

VARPTR(variable ou table ou n° fichier)

Donne l'adresse memoire d'une variable, d'une table ou de la mémoire tampon d'un fichier. Si l'adresse obtenue est négative, ajouter 65536.
Pour une variable numérique, la valeur est représentée à l'adresse obtenue sur 2,4 ou 8 octets suivant le type de la variable (%,!,#).
Le nom de la variable est représenté par les 2 octets avant l'adresse obtenue.
Ci-dessous, nous recherchons l'adresse d'une variable entière.

```
10 AA%=128
20 AD=VARPTR(AA%):IF AD<0 THEN AD=AD+65536!
30 PRINT "AD=";AD
50 FOR I=AD-2 TO AD+1:PRINT PEEK(I);:NEXT I

AD= 49252
 65  65  128  0
```

Ci-dessous, nous montrons que la représentation interne des variables double-précision est du type Décimal Code Binaire (DCB).
En effet 99 DCB → 153 binaire.

```
10 AA=99999999#
20 AD=VARPTR(AA)
30 IF AD<0 THEN AD=AD+65536!
40 FOR I=AD-3 TO AD+7:PRINT PEEK(I);:NEXT I

 8  65  65  72  153  153  153  153  0  0  0
```

99 décimal

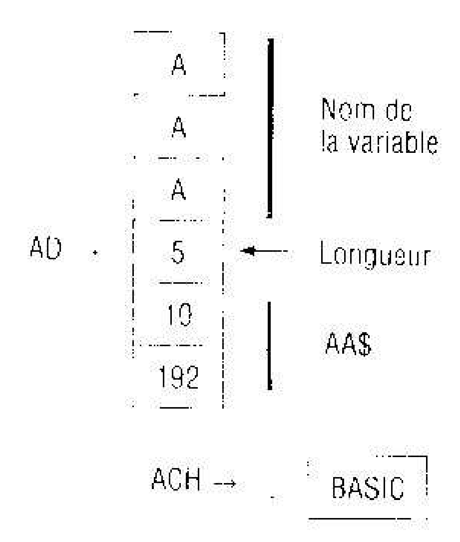
Pour obtenir l'adresse d'implantation d'une table numérique, il faut rechercher l'adresse de l'élément 0.

```
10 DIM BB%(2)
20 BB%(0)=128
30 BB%(1)=64
40 BB%(2)=32
50 AD=VARPTR(BB%(0)):IF AD<0 THEN AD=AD+65536!
60 PRINT "ad=";AD
70 FOR I=AD TO AD+15
80  PRINT PEEK(I);
90 NEXT I
100 PRINT
110 PRINT "ad=";VARPTR(BB%(2))+65536!
                                                    BB%(0)
                     Nom
ad= 49357
 0  0  0  2  66  66  9  0  1  3  0 128  0 64  32 0
ad= 49368
                                 Dimension
```

Sur l'exemple ci-dessous, nous recherchons l'implantation de la chaine "AAA$= « BASIC »".
A l'adresse donnée par VARPTR, nous trouvons le descripteur de la chaîne indiquant la longueur et l'adresse de la chaîne.

```
10 AA$="BASIC"
20 AD=VARPTR(AA$):IF AD<0 THEN AD=AD+65536!
30 PRINT "AD=";AD
40 PRINT "DS:",
50 FOR I=AD-2 TO AD+2:PRINT PEEK(I);:NEXT I
60 '
70 ACH=PEEK(AD+2)*256+PEEK(AD+1)
80 '
90 '----------------- edition de la chaine
100 PRINT "ACH=";ACH
110 FOR I=ACH TO ACH+PEEK(AD)
120  PRINT PEEK(I);
130 NEXT I
AD= 49426
DS: 65  65  5  10  192 ACH= 49162

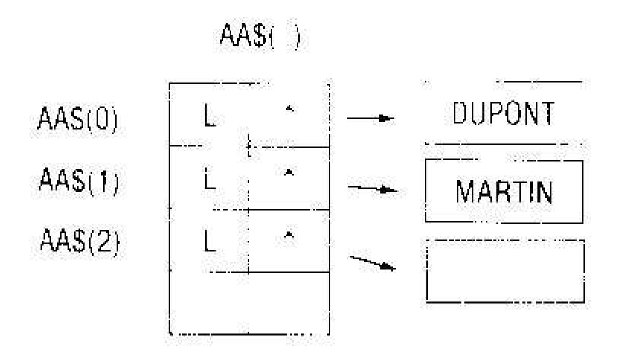
 66  65  83  73  67  34
```



A chaque fois qu'une chaîne change de longueur, elle est déplacée dans l'espace chaîne.

```
15 AA$ = AA$ + "X"
140 GOTO 15
```

L'adresse ACH de la chaîne change de valeur à chaque passage dans la boucle. Une table de chaîne est organisée ainsi :



VARPTR(AA$(0)) donne l'adresse du descripteur de la table de chaîne. Ce descripteur occupe trois octets par élément.

Mémoire écran :

La mémoire de l'écran est indépendante de la mémoire centrale. On y accède par VPEEK et VPOKE (au lieu de PEEK et POKE).

BASE(n)

Fournit les adresses des différentes parties de la mémoire écran.

n	
	SCREEN 0
0	Buffer texte (0)
2	Générateur de caractères (2048)
	SCREEN 1
5	Buffer texte (6144)
6	Table des couleurs (8192)
7	Générateur de caractères (0)
8	Affectation des sprites (6912)
9	Dessin des sprites (14336)
	SCREEN 2
10	Buffer texte (6144)
11	Table des couleurs (8192)
12	Générateur de caractères (6912)
13	Affectation des sprites (6912)
14	Dessin des sprites (14336)
	SCREEN 3
15	Buffer texte (2048)
17	Générateur de caractères (0)
18	Affectation des sprites (6192)
19	Dessin des sprites (14336)

```
10 FOR B=0 TO 19
20 PRINT B;BASE(B)
30 NEXT B

 0  0
 1  0
 2  2048
 3  0
 4  0
 5  6144
 6  8192
 7  0
 8  6912
 9  14336
 10  6144
 11  8192
 12  0
 13  6912
 14  14336
 15  2048
 16  0
 17  0
 18  6912
 19  14336
```

Ci-dessous, nous éditons la partie de la mémoire contenant la définition des couleurs (32 octets).

```
10 SCREEN1
20 CE=4:CF=15
30 COLOR CE,CF
40 FOR M=BASE(6) TO BASE(6)+32
50  PRINT VPEEK(M);
60 NEXT M
10 SCREEN1
20 CE=4:CF=15
30 COLOR CE,CF
40 FOR M=BASE(6) TO BA

 79  79  79  79  79  79  79  79  79  79  79  79  79  79  79  79  79
 79  79  79  79  79  79  79  79  79  79  79  79  255
```

Le code 79 représente : couleur écriture * 16 + couleur de fond.

VPEEK(adresse) :

Donne le contenu d'un octet de la mémoire écran.

```
10 SCREEN 0
20 PRINT VPEEK(2568)

RUN
32
```

2568 – →32
2569 – →80

Le contenu de la mémoire 2568 donne la valeur 32 qui correspond au premier des 8 octets représentant le caractère A (voir chapitre "Redéfinition des caractères").

VPOKE adresse,valeur

Modifie le contenu d'un octet de la mémoire écran.

En frappant :

```
SCREEN 0
VPOKE 150,65
```

vous voyez le caractère "A" s'afficher au milieu de l'écran.

Si vous frappez :

VPOKE 2568,0

le caractère "A" est modifié.

VDP(n)

Permet de lire les 8 registres du Vidéo Display Processor.

```
10 FOR N=0 TO 7:PRINT VDP(N):NEXT N
```

ENTRÉES-SORTIES DIRECTES

- INP
- OUT
- WAIT

INP(n° port)

Lit directement une entrée de périphérique.
"**n° port**" doit être compris entre −32768 et 65535. La valeur lue est comprise entre 0 et 255.

OUT n° port,valeur

Envoie une valeur vers un périphérique. La ligne ci-dessous allume la lampe de la touche "CAPS".

```
10 OUT (&HAA),INP(&HAA) AND &HBF
```

WAIT n° port,octet1,octet2

Suspend l'exécution du programme. L'exécution se poursuit lorsque(entrée port) AND (octet1)<>0
Si "**octet2**" est spécifié, un XOR modifie l'entrée avant l'opération AND avec "**octet1**".

TRAITEMENT DES ERREURS | 8

LA MISE AU POINT DES PROGRAMMES

- CTRL/STOP
- CONT
- STOP
- TRON
- TROFF

Les programmes ne fonctionnent pas toujours "du premier coup". BASIC envoie des messages pour certaines erreurs (de syntaxe par exemple) mais ne détecte pas les erreurs de logique. Pour les cas les plus délicats, il faut suivre le déroulement du programme étape par étape, ce qui est relativement simple en BASIC.

CTRL/STOP

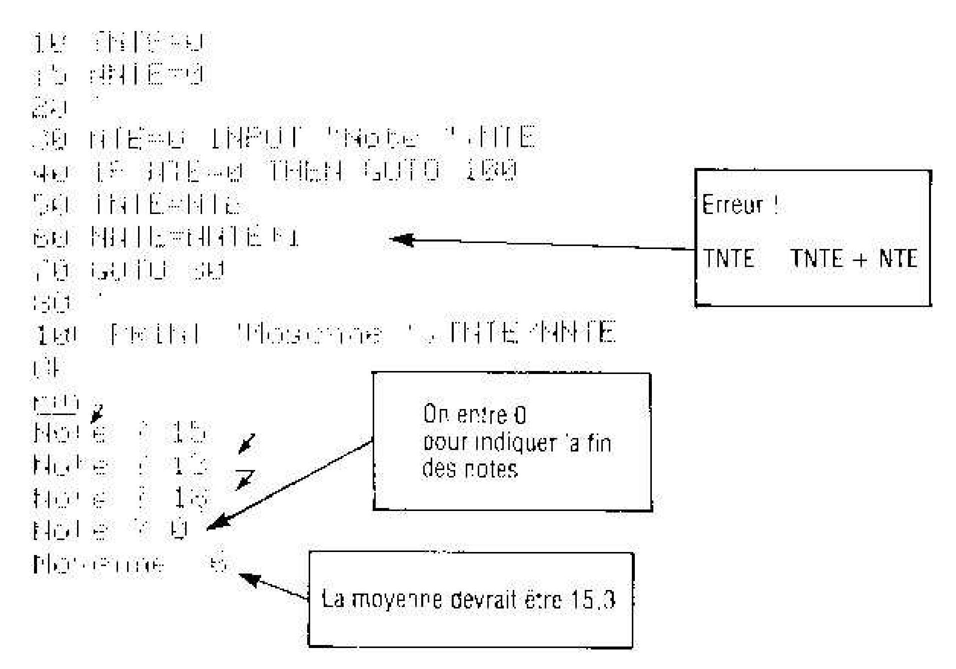
En appuyant sur CTRL STOP nous interrompons l'exécution du programme. Nous pouvons alors visualiser les valeurs des variables en mode immédiat.

CONT

L'exécution interrompue peut être poursuivie en frappant CONT (continue).

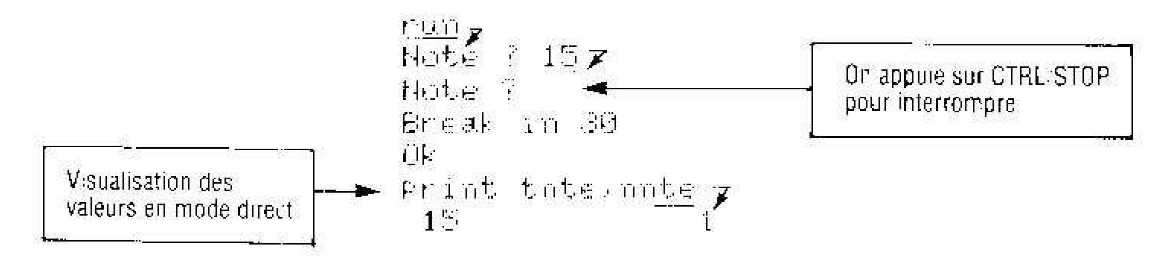
Exemple :

Le programme ci-dessous effectue la moyenne de plusieurs notes. Nous avons commis (volontairement !) une erreur. En 50, au lieu de TNTE = TNTE + NTE, nous avons écrit TNTE = NTE.



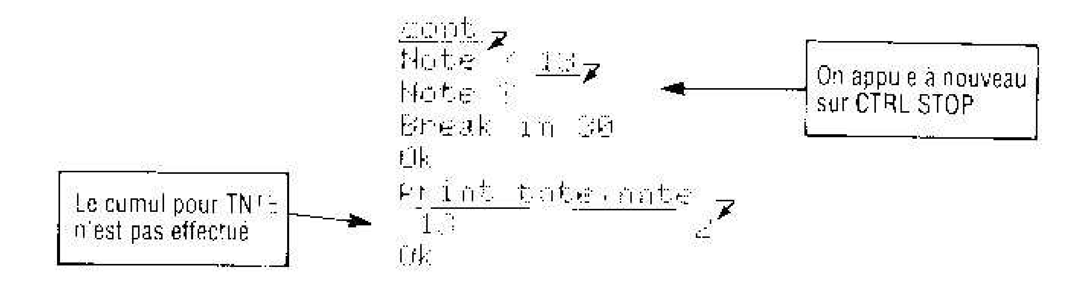
Remarque : Les instructions 10 et 15 qui initialisent les valeurs de TNTE et NNTE à zéro ne sont pas indispensables puisque RUN les initialise à zéro. Il est cependant plus prudent de le faire dans un programme plus important où TNTE et NNTE pourraient dejà avoir été utilisées dans une autre partie de programme et avoir une valeur non nulle.

Exécutons à nouveau le programme et interrompons-le après avoir entré la première note. Nous pouvons visualiser en mode direct les valeurs des variables NTE, TNTE et NNTE.



Pour l'instant, rien d'anormal.

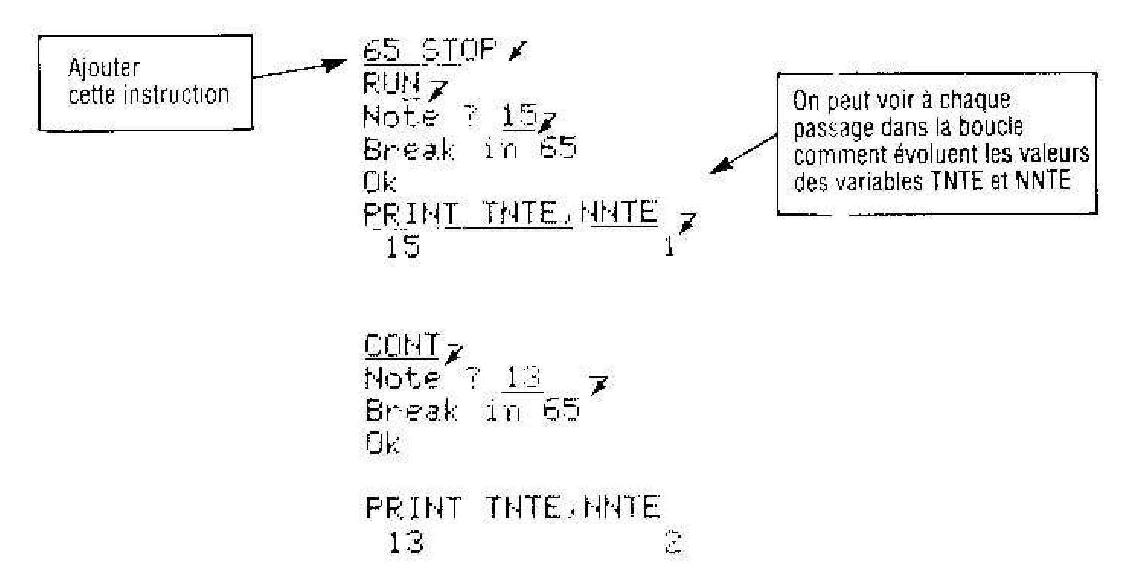
Frappons CONT pour continuer l'exécution du programme et interrompons à nouveau le programme après avoir entré la 2ᵉ note :



Nous nous apercevons en regardant la valeur de TNTE que le cumul des notes n'est pas effectué.

STOP

Au lieu d'appuyer sur CTRL/STOP nous aurions pu placer une instruction "STOP" en 65.



En cas de "BOUCLAGE" de programme :

Lorsqu'un programme ne s'arrête pas, appuyer sur CTRL/STOP, frapper TRON puis CONT.
On peut ainsi localiser la partie du programme où la boucle s'effectue.

```
10 I=1
20 '
30 I=I+1
40 IF I=5 THEN 70
50 GOTO 40
60 '
70 END
Ok
TRON
Ok
RUN
[10][20][30][40][50][40][50][40][50][
40][50][40][50][40][50][40][50][40][5
0][40][50][40][50][40][50][40][50][40
][50][40][50][40][50][40][50][40][50]
[40][50][40][50][40][50][40][50][40][
50][40]
Break in 40
Ok
TROFF
```

TRON-TROFF

Les commandes "TRON" et "TROFF" peuvent être inclues comme instructions dans un programme pour visualiser les numéros d'instructions dans la partie suspecte du programme.

LE TRAITEMENT D'ERREURS

- ERR
- ERL
- ON ERROR GOTO
- RESUME
- ERROR

ERR-ERL

Dès que survient une erreur pendant l'exécution d'un programme et si l'instruction "ON ERROR GOTO N° LIGNE" a été prévue, il y a branchement à un programme d'erreur au numéro de ligne spécifié dans l'instruction "ON ERROR GOTO".
Ce programme d'erreur analyse alors l'erreur en testant les valeurs de ERR et ERL qui représentent respectivement le code erreur et le numéro de ligne où s'est produite l'erreur
Après avoir analysé et traité l'erreur, l'instruction RESUME permet au programme d'erreur de provoquer un retour au programme où s'était produite l'erreur.

```
10 ON ERROR GOTO 100        ' en cas d'erreur
15 '
20 INPUT "Diviseur ";D
30 PRINT 10/D
40 GOTO 20
50 '
90 '-------------- analyse erreur
100 PRINT "Err=";ERR;"ERL=";ERL
105 IF ERR=11 AND ERL=30 THEN PRINT "Div Par 0 interdite":RESUME 20
110 PRINT "Erreur non reconnue":STOP
 Ok
 run
 Diviseur ? 0
 Err= 11 ERL= 30
 Div Par 0 interdite
 Diviseur ?
```

ON ERROR GOTO 0

Ecrit dans un programme de traitement d'erreur, il annule ON ERROR GOTO N° ligne. L'erreur est donc traitée normalement par le système (interruption du programme et message d'erreur).

RESUME

Ecrit à la fin d'un programme de traitement d'erreur, RESUME spécifie où doit se poursuivre l'exécution du programme :

RESUME	l'exécution se poursuit au numéro de ligne où s'est produite l'erreur.
RESUME NEXT	l'exécution se poursuit au numéro de ligne après celui où s'est produite l'erreur.
RESUME N° ligne	l'exécution se poursuit au numéro de ligne spécifié.

ERROR N° erreur

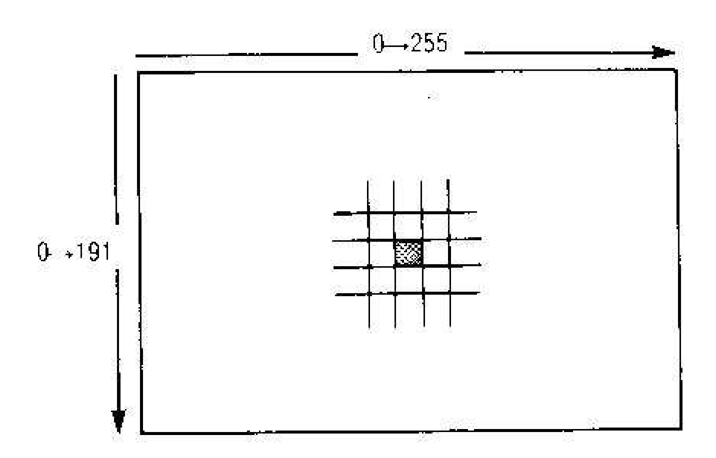
Permet à l'utilisateur de définir ses propres codes erreurs (compris entre 0 et 255) et de provoquer un branchement à ON ERROR GOTO... comme si une erreur avait eu lieu.

GRAPHISMES ET SONS | 9

LE GRAPHIQUE HAUTE RÉSOLUTION

- PSET
- PRESET
- LINE
- CIRCLE
- PAINT
- POINT
- DRAW

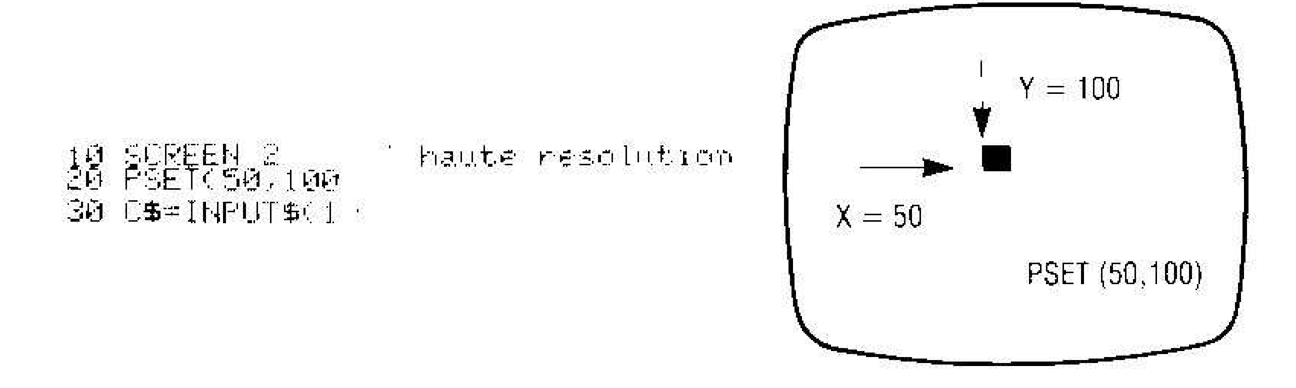
En haute résolution (SCREEN 2), l'écran est divisé en 256×192 points.



A la fin d'un programme utilisant la haute résolution il faut placer une instruction d'attente afin d'éviter un retour au mode SCREEN0 ou SCREEN1 ce qui effacerait l'écran graphique.
Pour que la couleur de fond choisie par COLOR soit prise en considération, elle doit être programmée AVANT l'instruction SCREEN2.

PSET (X,Y),couleur
PSET STEP (DX,DY),couleur

PSET(X,Y) allume le point X,Y.
PSET STEP (DX,DY) spécifie des déplacements relatifs au point courant. Par défaut, la couleur est celle définie par COLOR.
Le programme ci-dessous allume le point à l'intersection de la colonne 50 et de la ligne 100.



```
10 SCREEN 2        ' haute resolution
20 PSET(50,100)
30 C$=INPUT$(1)
```

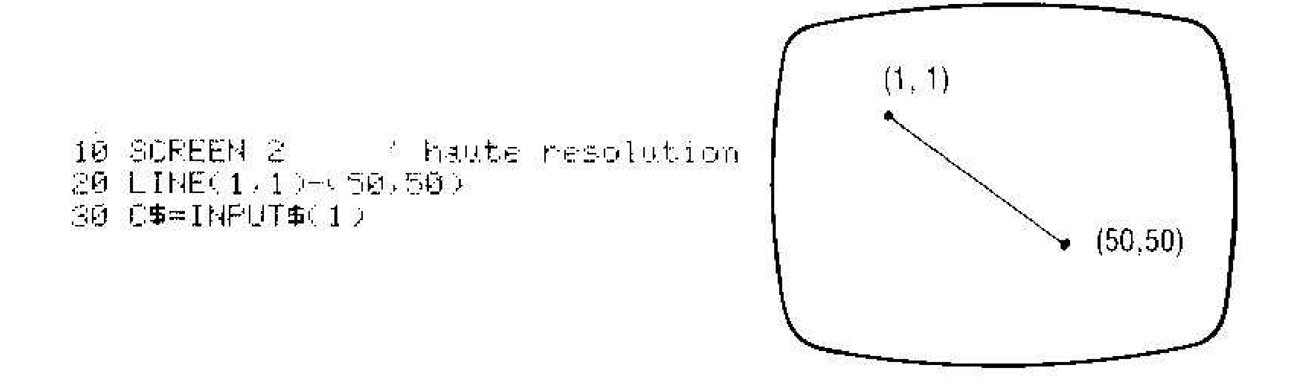

L'instruction 30 bloque le programme en haute résolution en attendant que l'opérateur appuie sur une touche quelconque.

PRESET (X,Y)
PRESET STEP (DX,DY)

Donne la couleur de fond au point spécifié.

LINE (Xdep,Ydep)–(Xfin,Yfin),couleur
LINE STEP (DX1,DY1)–STEP (DX2,DY2),couleur

Trace une droite entre les points spécifiés. STEP spécifie des déplacements relatifs. La couleur par défaut est celle spécifiée par COLOR.

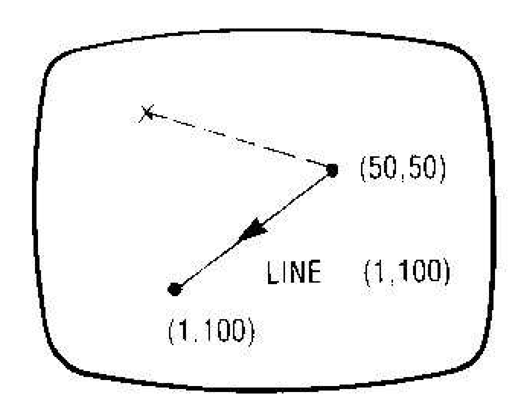


```
10 SCREEN 2        ' haute resolution
20 LINE(1,1)-(50,50)
30 C$=INPUT$(1)
```

LINE −(Xfin,Yfin),couleur

Trace une droite à partir du point courant.

```
10 SCREEN 2      ' haute resolution
20 LINE(1,1)-(50,50)
25 LINE -(1,100)
30 C$=INPUT$(1)
```



LINE (Xdep,Ydep)−(Xfin,Yfin),couleur,B ou BF

Trace une boîte. Les points donnés sont ceux du sommet en haut à gauche et celui du sommet en bas à droite.
"**B**" spécifie une boîte vide, "**BF**" une boîte pleine.
Naturellement, l'option STEP est acceptée.

```
10 SCREEN 2                    ' haute resolution
20 LINE(1,1)-(50,50),4,B       ' boite bleue
30 C$=INPUT$(1)
```

Exemples avec STEP :

STEP spécifie des déplacements relatifs au point courant.
Le programme ci-dessous trace une droite entre les points (100.100) et (110,110).

```
10 SCREEN 2                ' haute resolution
15 PSET(100,100)
20 LINE -STEP(10,10)       ' deplacement relatif
30 C$=INPUT$(1)
```

Celui-ci trace une droite entre (110,110) et (120,120).

```
10 SCREEN 2
20 PSET (100,100)
30 LINE STEP(10,10)-STEP(20,20
40 C$=INPUT$(1)
```

Le programme suivant trace un dessin défini en relatif. Ceci permet un tracé avec échelle.

```
10 '------------------ trace de dessin en relatif
20 SCREEN 2
30 DATA 5,0,  0,8,  -5,0,  0,-8, 999,999
40 '
50 PSET(100,100) ECH=1:GOSUB 120   ' echelle 1
60 RESTORE
70 PSET (100,150) ECH=2:GOSUB 120 ' echelle 2
80 C$=INPUT$(1)
90 END
100 '
110 '------------------ sous-Programme
120 READ DX,DY:IF DX=999 THEN RETURN
130 LINE -STEP(ECH*DX,ECH*DY)
140 GOTO 120
```

Tracé du drapeau français.

```
20 COLOR 4,14:SCREEN 2       ' haute resolution
30 XA=100:YA=40:H=30:L=20
40 PSET (XA,YA)
50 LINE STEP(0,0)-STEP(L,H),4,BF      ' bleu
60 LINE STEP(0,-H)-STEP(L,H),15,BF   ' blanc
70 LINE STEP(0,-H)-STEP(L,H),6,BF    ' rouge
80 C$=INPUT$(1)
```

CIRCLE (XC,YC),rayon,couleur

Dessine un cercle dont le centre est XC,YC. Si la couleur est omise, le cercle est dessiné avec la couleur courante.

```
10 SCREEN 2
20 CIRCLE (100,100),50    ' centre/rayon
30 C$=INPUT$(1)
```

CIRCLE (XC,YC),rayon,couleur,angle début,angle fin,rapport
CIRCLE STEP (DX,YX),rayon,couleur,angle début,angle fin, rapport

Permet de tracer des arcs de cercle.
"**rapport**" permet d'obtenir des ellipses.
Pour corriger l'aspect des cercles sur les téléviseurs français, on fera : "rapport"=1.3.

```
5 '-------------- arc en ciel
10 SCREEN 2                   ' haute resolution
20 PI=3.14159
30 FOR N=1 TO 7
40 CIRCLE (100,100),20+N*8,N,0,PI,1.3
50 NEXT N
60 C$=INPUT$(1)
```

PAINT (X,Y),couleur,bordure
PAINT STEP(DX,DY),couleur,bordure

Remplit une figure dans la couleur spécifiée. La couleur spécifiée doit être celle de la figure.
"**bordure**" spécifie en BASSE RÉSOLUTION la couleur qu'aura la figure remplie.

```
10 '----- Cercle Plein
20 CE=2:CF=15                 ' ecriture'fond
30 COLOR CE,CF:SCREEN 2
40 '
50 X=50:Y=50                  ' centre
60 R=20                       ' rayon
70 CIRCLE (X,Y),R
80 PAINT (X,Y),CE             ' remplissage
90 C$=INPUT$(1)
```

■ **Cocarde tricolore :**

```
10 '------------- cocarde tricolore
20 COLOR 1,15:SCREEN 2
30 CIRCLE (50,50),30,6       ' rouge
40 PAINT(50,50),6
50 CIRCLE (50,50),20,15      ' blanc
60 PAINT(50,50),15
70 CIRCLE (50,50),10,4       ' bleu
80 PAINT(50,50),4
90 C$=INPUT$(1)
```

■ **Arc-en-ciel avec PAINT :**

```
5 '------------------- arc en ciel
10 COLOR 1,15:SCREEN 2
20 PI=3.14159
30 FOR N=9 TO 1 STEP-1
40  CIRCLE(100,100),20+N*8,N,0,PI,1.3
50  LINE (1,100)-(200,100),N
60  PAINT(100,100-2),N
70 NEXT N
80 C$=INPUT$(1)
```

POINT (X,Y)

Fournit la couleur d'un point.

```
10 CLS
20 PSET(4,100),4
30 PRINT POINT(4,100)
RUN
 4
```

■ Recopie d'écran en haute résolution (imprimante Seikosha)

Ce programme recopie l'écran sur une imprimante "SEIKOSHA". Son exécution est lente.

```
1000 '----- RECOPIE ECRAN HAUTE RESOLUTION IMPRIMANTE SEIKOSHA
1005 '
1010 DIM L(250)
1020 CF=15                        ' couleur de fond (15:à adapter)
1025 Y=0
1030 FOR N=1 TO 24                ' 24 lignes
1040  FOR I=1 TO 250:L(I)=128:NEXT I
1050  FOR M=1 TO 7
1060   FOR X=0 TO 249
1070    P=POINT(X,Y):IF P<>CF THEN L(X)=L(X)+2^(M-1)
1080  NEXT X
1090  Y=Y+1
1100 NEXT M
1110 '----- edition 1 ligne
1120 LPRINT CHR$(8);
1130  FOR I=1 TO 250
1140   LPRINT CHR$(L(I));
1150  NEXT I
1160  LPRINT
1170 NEXT N
```

Ce programme pourra être sauvegardé par : **SAVE "RECOP"** plutôt que par CSAVE. Ainsi, il pourra être ajouté à un autre programme par "**MERGE "RECOP"**".

GESTION DES COULEURS :

En principe, la couleur peut être spécifiée pour chaque point. En réalité, BASIC gère les couleurs de l'écran par groupes horizontaux de 8 points.
A l'intérieur d'un groupe de 8 points, il ne peut y avoir plus de 2 couleurs différentes.
Par exemple, pour un fond blanc, si vous faites "PSET(17,4),4" le point (17,4) devient bleu.
Si maintenant, vous faites "PSET(20,4),6" le point précédent prend la couleur du point (20,4), c'est-à-dire rouge.




```
10 COLOR 1,15:SCREEN 2
20 PSET(17,4),4                 'bleu
30 FOR TP=1 TO 1000:NEXT TP
40 PSET(20,4),6                 'rouge
50 C$=INPUT$(1)
```

L'intersection d'une droite horizontale et d'une droite verticale ne pose pas de problème puisqu'à l'intersection, il n'existe que 2 couleurs (pas de couleur de fond).

```
10 '------------------- intersection de droites
20 COLOR 1,15
30 SCREEN 2
40 '------------------------------- Pas de Probleme
50 LINE(1,10)-(30,10),14
60 LINE (20,5)-(20,20),1
70 '--------------------------------- Probleme
80 LINE(100,110)-(150,120),14
90 LINE(100,130)-(150,110),1
100 C$=INPUT$(1)
```

La superposition de figures pleines de couleurs différentes ne pose pas de problème, pourvu qu'il n'y ait pas plus de deux couleurs dans un groupe horizontal de 8 points.
Pour définir plusieurs couleurs de fond sur un même écran, utiliser :

LINE (X1,Y1)—(X2,Y2),couleur,BF

```
10 '------------ couleur de fond
20 COLOR 1,15:SCREEN 2
30 LINE (1,1)-(100,100),14,BF      ' fond gris
40 LINE (100,1)-(200,100),6,BF     ' fond rouge
50 CIRCLE(50,50),30,1
60 CIRCLE(150,50),30,1
70 C$=INPUT$(1)
```

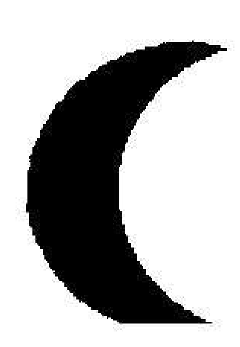
Nous présentons ci-dessous divers programmes :

■ **Anneau :**

```
10 '------ Anneau
20 CE=2:CF=15              ' ecriture'fond
30 COLOR CE,CF:SCREEN 2
40 '
50 X=50:Y=50               ' centre
60 R2=30:R1=20             ' rayons
70 CIRCLE (X,Y),R2
80 CIRCLE (X,Y),R1
90 PAINT (X,Y+R1+1),CE
100 C$=INPUT$(1)
```

■ Croissant :

```
20 CF=15                          ' couleur de fond
30 COLOR 1,CF
40 SCREEN 2                       ' haute resolution
50 CIRCLE(100,100),50,1
60 PAINT(100,100),1               ' 1er cercle plein
70 '
80 CIRCLE (130,100),50,CF         ' 2eme cercle plein couleur de fond
90 PAINT (130,100),CF
100 C$=INPUT$(1)
110 END
```

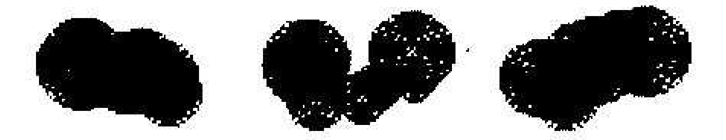


■ Boîtes aléatoires :

```
10 '------------- trace de boites aleatoires
20 COLOR 1,15:SCREEN 2
40 '
50 FOR N=1 TO 30                  ' 30 boites
60  X=RND(1)*220:Y=RND(1)*170
70  H=RND(1)*15+5:L=RND(1)*15+5
80  CL=RND(1)*15
90  LINE(X,Y)-(X+L,Y+H),CL,BF
100 NEXT N
110 '
120 C$=INPUT$(1)
```

■ Nuages :

```
10 '------------ NUAGES
20 COLOR 1,15:SCREEN 2
40 FOR NU=1 TO 3                    ' 3 nuages
50   X0=(NU-1)*70+40:Y0=50+RND(1)*10
60   C=RND(1)*14+1                  ' couleur
70   FOR N=1 TO 10                  ' 10 cercles pour 1 nuage
80    X=RND(1)*40+X0
90    Y=RND(1)*20+Y0
110   R=5+RND(1)*10
120   FOR R1=1 TO R:CIRCLE (X,Y),R1,C:NEXT R1
130  NEXT N
140 NEXT NU
150 C$=INPUT$(1)
```



■ Télécran haute résolution :

Vous déplacez un curseur clignotant qui laisse une "trace" sur son passage. Huit directions ont été prévues. Si vous "levez" le curseur avec "L", le curseur est déplacé sans écriture. "B" permet de "baisser" le curseur. En frappant "E" et en déplaçant le curseur, vous effacez.

```
10 '---------- TELECRAN HAUTE RESOLUTION (8 DIRECTIONS+COULEURS)
20 '
30 CF=15:CL=1                    ' couleur fond et écriture
40 COLOR CL,CF
50 SCREEN 2
60 '
70 OPEN "GRP:" FOR OUTPUT AS #1
80 PSET(14,160):PRINT #1,"CLAVIER MAJUSCULE"
90 PSET(14,170)
100 PRINT #1,"FLECHES ET SWAS / 1,2..COULEURS"
110 PSET(14,180)
120 PRINT #1,"L:LEVER/ B:BAISSER  E:EFFACER"
130 X=100:Y=100                    ' COORDONNEES DEPART
140 '----------------- curseur clignotant
150 T=POINT(X,Y)
160 '
170 C$=INKEY$:IF C$<>"" THEN 220 ' test clavier
180 PSET(X,Y),CL
190 PSET(X,Y),CF
200 GOTO 170
210 '-------
220 IF L=0 THEN PSET(X,Y),CL
230 IF L=1 THEN PSET(X,Y),T
240 IF L=2 THEN PSET(X,Y),CF
250 '
260 C=ASC(C$)
270 IF C=29 THEN X=X-1           ' gauche
280 IF C=28 THEN X=X+1           ' droite
```

→

```
290 IF C=31 THEN Y=Y+1                ' bas
300 IF C=30 THEN Y=Y-1                ' haut
310 IF C$="Q" THEN X=X-1:Y=Y-1        ' gauche/haut
320 IF C$="W" THEN X=X+1:Y=Y-1
330 IF C$="A" THEN X=X-1:Y=Y+1
340 IF C$="S" THEN X=X+1:Y=Y+1
350 '
360 IF X<1 THEN X=1
370 IF X>254 THEN X=254
380 IF Y<1 THEN Y=1
390 IF Y>180 THEN Y=180
400 '-
410 IF C$="L" THEN L=1               ' lever
420 IF C$="B" THEN L=0               ' baisser
430 IF C$="E" THEN L=2               ' effacer
440 IF VAL(C$)>0 AND VAL(C$)<10 THEN CL=VAL(C$)     ' couleur
450 GOTO 150
```

Le système "télécran" n'est pas très puissant. On regardera, dans la partie "Programmes" les programmes "Tracé par segments de droite" et "Dessinateur".

■ Ce programme trace des **figures inscrites dans un cercle :**

```
10 '---------------- figures inscrites dans un cercle
20 COLOR 4,15:SCREEN 2
30 '
40 PI=3.14159
50 XC=100:YC=100                ' centre
60 R=20:CL=4                    ' rayon/couleur
70 NCOT=3                       ' nombre de cotes
80 GOSUB 170
90 '---------
100 NCOT=6:XC=150:R=30:CL=4:GOSUB 170 ' hexagone
110 '---
120 NCOT=15:R=30:YC=40:GOSUB 170       ' cercle
140 C$=INPUT$(1)
150 '------------------------------- SPGM
170 PSET(XC+R,YC),CL        ' Premier Point
180 FOR C=1 TO NCOT
190   A=C*PI*2/NCOT
200   X=XC+R*COS(A)
210   Y=YC+R*SIN(A)
220   LINE -(X,Y),CL
230 NEXT C
240 RETURN
```

■ Le programme ci-dessous trace des **étoiles** dont on a défini le nombre de côtés (5,7,9,...) :

```
10 '------------ TRACE D'ETOILES
20 COLOR 4,15 SCREEN 2
30 '
40 PI=3.14159
50 XC=100:YC=100            ' centre
60 R=20:CL=4                ' rayon/couleur
70 NCOT=7                   ' nombre de branches
80 GOSUB 160
90 '------------
100 NCOT=5:XC=150:R=30:CL=4:GOSUB 160
110 '---
120 NCOT=15 R=30:YC=40 GOSUB 160
130 C$=INPUT$(1)
140 END
150 '------------------------------ SPGM
160 PSET(XC+R,YC),CL              ' Premier Point
170 FOR C=1 TO NCOT
180    A=C*(NCOT-1)*PI/NCOT
190    X=XC+R*COS(A)
200    Y=YC+R*SIN(A)
210    LINE -(X,Y),CL
220 NEXT C
230 RETURN
```

■ **Histogramme circulaire :**

```
10 '------------------------------histogramme circulaire
30 H(1)=.2:H$(1)="tarte"
40 H(2)=.3 H$(2)="croissant"
50 H(3)=.1:H$(3)="brioche"
60 H(4)=.3:H$(4)="eclair"
70 H(5)=.1 H$(5)="Pain"
80 '
90 XA=120 YA=100
100 R=30
110 PI=3.14159
120 '--
130 COLOR 4,15:SCREEN 2
135 OPEN "GRP:" FOR OUTPUT AS #1
140 CIRCLE (XA,YA),R
150 AA=0
160 FOR P=1 TO 5
170  A=AA+PI*2*H(P)
180  X=XA+R*COS(A):Y=YA+R*SIN(A)
190  LINE (XA,YA)-(X,Y)
200 '
210  AT=AA+PI*H(P)           ' affichage texte
220  X=XA+R*1.3*COS(AT):Y=YA+1.3*R*SIN(AT)
230  IF AT>PI/2 AND AT<3*PI/2 THEN X=X-8*LEN(H$(P))
240  DRAW "BM=X;,=Y;":PRINT #1,H$(P)
250  AA=A
260 NEXT P
270 C$=INPUT$(1)
```

■ Chronomètre :

```
10 '---------- ---- chronometre
20 CE=15:CF=4                    ' ecriture/fond
30 COLOR CE,CF:SCREEN 2
40 PI=3.14159
50 XA=100:YA=80:R=50             ' centre/rayon
60 CIRCLE (XA,YA),R
70 FOR A=0 TO 2*PI STEP PI/6     ' graduations
80   X=XA+R*.8*COS(A):Y=YA+R*.8*SIN(A)
90   XB=XA+R*COS(A):YB=YA+R*SIN(A)
100 LINE (X,Y)-(XB,YB)
110 NEXT A
120 '---------- --- -- --- mouvement
130 R=R*.7
140 FOR A=0 TO 100 STEP PI/30
150   X=XA+R*COS(A):Y=YA+R*SIN(A)
160   LINE (XA,YA)-(X,Y)
170   BEEP
180   FOR TP=1 TO 200:NEXT TP          ' temporisation
190   LINE (XA,YA)-(X,Y),CF            ' effacement
200 NEXT A
210 C$=INPUT$(1)
```

DRAW chaîne de caractères

Cette instruction permet de représenter des formes plus rapidement que l'instruction LINE.

■ Déplacements relatifs

Une forme est représentée par des déplacements relatifs à la position courante du curseur.

Par exemple, pour représenter un carré de 20×20, nous codons :

R20 '20 positions à droite
D20 '20 positions vers le bas
L20 '20 positions à gauche
U20 '20 positions vers le haut

```
10 SCREEN 2
20 PSET(100,100)              ' Positionnement
30 DRAW "R20D20L20U20"        ' carre
40 C$=INPUT$(1)
```

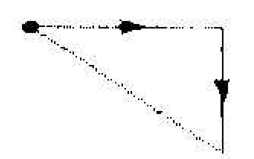
Les huit directions sont :

U: haut	E: haut/droite
D: bas	F: bas/droite
R: droite	G: bas/gauche
L: gauche	H: haut/gauche

■ Coordonnées absolues : Mx,y

La commande "**Mx,y**" spécifie des coordonnées absolues.

```
10 SCREEN 2
20 PSET (100,100)          ' Positionnement
30 DRAW "M130,100M130,120M100,100" ' triangle
40 C$=INPUT$(1)
```



■ Déplacements relatifs : M+x,+y

La présence des signes +/− indique que les valeurs spécifiées sont des déplacements relatifs.

```
10 SCREEN 2
20 PSET (100,100)          ' Positionnement
30 DRAW "M+30,+0M+0,+20M-30,-20"  ' triangle
40 C$=INPUT$(1)
```

■ Déplacement sans écriture : B

Si "**B**" est spécifié, la première commande qui suit est effectuée sans écriture.

```
10 SCREEN 2
15 PSET(100,100)
20 DRAW "R20BU20L20"
100 C$=INPUT$(1)
```

L20

U20

R20

■ Retour à la position précédente : N

"**N**" provoque, après exécution de la commande suivante, le retour à la position avant la commande.

```
10 SCREEN 2
20 PSET(100,100)
30 A$="NU10NR10NL10ND10"
40 DRAW A$
50 C$=INPUT$(1)
```

■ Angle : A

"**A**" provoque la rotation d'un dessin.

A0: 0 degré
A1: 90 degrés
A2: 180 degrés
A3: 270 degrés

Ci-dessous, nous représentons un triangle dans quatre positions.

```
10 SCREEN 2
20 PSET(100,100)
30 A$="R20D10M-20,-10"
40 FOR I=0 TO 3
50  DRAW "A=I;XA$;"
60 NEXT I
100 C$=INPUT$(1)
```

■ Couleur : C

La couleur spécifiée devient la couleur courante.

```
10 SCREEN 2
20 PSET(100,100)
30 A$="R20D10M-20,-10"     ' triangle
40 DRAW "C2"+A$            ' couleur 2
50 PSET (150,100)
60 DRAW "C6"+A$            ' couleur 6
70 C$=INPUT$(1)
```

■ Echelle : S

"S" permet d'afficher le dessin avec un facteur d'échelle allant de 1 à 255. L'échelle s'obtient en divisant par 4 le facteur d'échelle.

```
10 SCREEN 2
20 PSET(100,100)
30 A$="R20D10M-20,-10"     ' triangle
40 DRAW "S4"+A$            ' echelle 1
50 PSET (150,100)
60 DRAW "S8"+A$            ' echelle 2
70 C$=INPUT$(1)
```

■ Passage de variables chaînes : Xchaîne

Pour spécifier des variables chaînes dans une chaîne, il suffit de placer **"X"** devant la variable chaîne et un point-virgule après.

```
10 SCREEN 2
15 '-------
20 PSET(100,100)
30 B$="R40U40"
40 C$="L40"
50 DRAW B$+C$
55 '-----
60 PSET(150,100)
70 DRAW "XB$;XC$;"
80 C$=INPUT$(1)
```

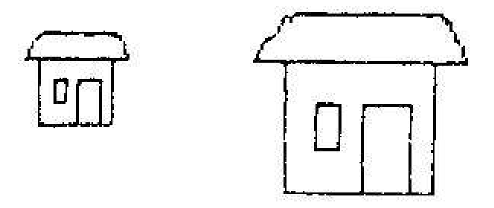
■ Passage de variables numériques

Les variables numériques sont spécifiées en ajoutant "=" devant le nom de la variable et point-virgule après.

```
10 SCREEN 2
20 X=100:Y=100
30 DRAW "BM=X;,=Y;"
40 DRAW "R20D20L20U20"
100 C$=INPUT$(1)
```

```
10 SCREEN 2
20 X=100:Y=100:HT=20:LG=30
25 DRAW "BM=X;,=Y;R=LG;D=HT;L=LG;U=HT;"
100 C$=INPUT$(1)
```

Les dessins ci-dessous ont été obtenus avec le générateur de formes présenté plus loin.



```
590 '------------------ MAISON
600 COLOR 4,15:SCREEN 2
610 D$="R27M-3,-8L19M-5,+8R4D18R19U18L9BD6D12R6U12L6BL6D6R3U6L3"
620 '------------------------ echelle 1
630 DRAW "BM100,80"+"S4"+D$
640 '------------------------ echelle 2
650 DRAW "BM160,80"+"S8"+D$
660 C$=INPUT$(1)
```

```
10 '------------------ bateau
20 COLOR 4,15:SCREEN 2,2
30 D$="R53BM-26,-40D37R26M-26,-37BL3D
37L18BM+18,-37M-19,+37BM-5,+3M+8,+12R
37M+8,-12"
40 '--------echelle 1
50 DRAW "BM100,80"+"S4"+D$
70 '------------- echelle 3/4
80 DRAW "BM100,120"+"S3"+D$
90 C$=INPUT$(1)
```

GÉNÉRATEUR DE FORMES :

Vous réalisez un dessin par segments de droite. Le programme affiche la chaîne D$ représentant le dessin.

```
10 '----------- GENERATEUR DE FORME AVEC DRAW
20 '
30 CF=15:CE=1                    ' couleur fond et ecriture
40 COLOR CE,CF:SCREEN 2
50 '
60 OPEN "GRP:" FOR OUTPUT AS #1
70 PSET(14,150):PRINT #1,"CLAVIER MAJUSCULE"
80 PSET(14,160):PRINT #1,"1ER POINT:FLECHES PUIS 'V'"
90 PSET(14,170):PRINT #1,"AUTRES POINTS: T:TRACE DROITE"
100 PSET(14,180):PRINT #1,"L:LEVER/ B:BAISSER F:FIN"
110 X=100:Y=80                 ' coordonnees depart
120 SPRITE$(1)=CHR$(191)       ' curseur
130 '-------------------- curseur
140 PUT SPRITE 1,(X,Y-1),1,1
150 '
160 C$=INKEY$:IF C$="" THEN 160 ' test clavier
170 '
180 C=ASC(C$)
190 IF C=29 THEN X=X-1:GOTO 140 ' gauche
200 IF C=28 THEN X=X+1:GOTO 140 ' droite
210 IF C=31 THEN Y=Y+1:GOTO 140 ' bas
220 IF C=30 THEN Y=Y-1:GOTO 140 ' haut
230 IF C$="V" THEN PSET(X,Y),CE:XA=X:YA=Y:GOSUB 310
240 IF C$="T" AND L=0 THEN LINE (XA,YA)-(X,Y),CE
250 IF C$="T" THEN GOSUB 360:XA=X:YA=Y:PSET(X,Y),CE
260 IF C$="F" THEN PSET(14,10):PRINT #1,D$:LPRINT D$
270 IF C$="L" THEN L=1          ' lever
280 IF C$="B" THEN L=0          ' baisser
290 GOTO 140
300 '-------------- 1ER POINT
310 X$=STR$(X):X$=RIGHT$(X$,LEN(X$)-1)
320 Y$=STR$(Y):Y$=RIGHT$(Y$,LEN(Y$)-1)
330 D$=D$+"BM"+X$+","+Y$
340 RETURN
350 '-------------------- AUTRES POINTS
360 IF L=1 THEN D$=D$+"B"
370 DX=X-XA:DY=Y-YA
380 DX$=STR$(DX):DX$=RIGHT$(DX$,LEN(DX$)-1)
390 DY$=STR$(DY):DY$=RIGHT$(DY$,LEN(DY$)-1)
400 IF DX=0 AND DY>0 THEN D$=D$+"D"+DY$:RETURN
410 IF DX=0 AND DY<0 THEN D$=D$+"U"+DY$:RETURN
420 IF DY=0 AND DX>0 THEN D$=D$+"R"+DX$:RETURN
430 IF DY=0 AND DX<0 THEN D$=D$+"L"+DX$:RETURN
440 '
450 IF DX=>0 THEN DX$="+"+DX$
460 IF DX<0  THEN DX$="-"+DX$
470 '
480 IF DY=>0 THEN DY$="+"+DY$
490 IF DY<0 THEN DY$="-"+DY$
500 D$=D$+"M"+DX$+","+DY$
510 RETURN
```

MÉLANGE TEXTE/GRAPHIQUE :

Pour afficher du texte en mode graphique, il faut ouvrir un canal pour le pseudo-périphérique "GRP:".
Ensuite, on positionne le curseur avec PRESET(X,Y), PSET(X,Y) OU BMX,Y.
Enfin, on écrit le texte avec PRINT #,TEXTE.

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
30 DRAW "BM100,100":PRINT #1,"COUCOU"
100 C$=INPUT$(1)
```

Les instructions d'édition telles que "PRINT TAB" ne fonctionnent pas.
Un texte affiché sur un autre n'efface pas ce dernier. Pour effacer le premier texte, il faut :

- Réécrire le texte en couleur de fond.
- Afficher des "Pavés" en couleur de fond.

```
10 CE=4:CF=15:COLOR CE,CF          ' CF:couleur de fond
20 SCREEN 2
30 OPEN "GRP:" FOR OUTPUT AS #1
40 DRAW "BM100,100":PRINT #1,"COUCOU"
50 FOR TP=1 TO 2000:NEXT TP
60 '--effacement
70 COLOR CF:DRAW "BM100,100":PRINT #1,STRING$(10,CHR$(200))
80 '---
90 COLOR CE:DRAW "BM100,100":PRINT #1,"C'EST MOI"
100 C$=INPUT$(1)
```

■ Saisie d'une ligne en haute résolution

Nous proposons ici un programme permettant de saisir une ligne en haute résolution.

```
10 ' SAISIE D'UNE LIGNE EN HAUTE RESOLUTION
20 '
30 CE=1:CF=15
40 COLOR CE,CF
50 SCREEN 2
60 OPEN "GRP:" FOR OUTPUT AS #1
70 XL=10:YL=10:MES$="NOM:":GOSUB 100      ' coordonnees affichage
80 END
90 '----------------
100 LIG$=""
110 X1=(XL-LEN(MES$))*8:Y1=YL*8:DRAW "BM=X1;,=Y1;":PRINT #1,ME$
120 '
130 L=LEN(LIG$):X1=(XL+L)*8:Y1=YL*8
140 DRAW "BM=X1;,=Y1;"
150 '
160 C$=INPUT$(1)
```

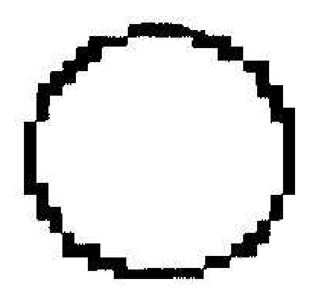
→

```
170 C=ASC(C$)
180 '
190 IF C<>8 THEN 220         ' code suppression
200 IF L>0 THEN LIG$=LEFT$(LIG$,L-1):COLOR CF:DRAW "L8":
PRINT #1,CHR$(200):COLOR CE:GOTO 130 ELSE 130
210 '
220 IF C=13 THEN 270         ' code de return
230 IF C<32 THEN BEEP:GOTO 130
240 LIG$=LIG$+C$             ' ajout caractere
250 PRINT #1,C$
260 GOTO 130
270 RETURN
```

LE GRAPHIQUE BASSE RÉSOLUTION

Les instructions en basse résolution sont les mêmes qu'en haute résolution.
Bien que l'écran soit divisé en 64*48 points, les coordonnées X,Y à spécifier sont les mêmes qu'en haute résolution (0 → 255) et (0 → 191). Seule la taille du point affiché est différente.
Ci-dessous, nous avons représenté un cercle.

```
10 SCREEN 3                   ' basse resolution
20 CIRCLE (50,50),30
30 C$=INPUT$(1)
```



Avec PAINT, pour obtenir une bordure de couleur différente, il faut que la couleur de bordure spécifiée dans PAINT soit la même que la couleur du cercle. La couleur de remplissage du cercle peut être quelconque.

```
5 '-------------- essai bordure en br
10 SCREEN 3                ' basse resolution
20 CIRCLE (100,100),40,2  ' cercle vert
30 PAINT (100,100),6,2    ' bordure verte
100 GOTO 100
```

■ **Télécran avec couleurs :**

Principe : En appuyant sur les flèches → ← ↑ ↓, vous déplacez un "point" qui laisse une "trace" sur son passage ce qui vous permet de dessiner "naturellement". Si vous appuyez sur "**L**", le déplacement s'effectue sans laisser de trace, permettant ainsi de dessiner une figure en plusieurs parties.
"**E**" permet d'effacer. Pour les déplacements en diagonale, se référer au programme "Télécran haute résolution".

```
10 '--------- TELECRAN BASSE RESOLUTION (4 DIRECTIONS+COULEURS)
20 '
30 CF=15:CL=1                    ' couleur fond et ecriture
40 COLOR CL,CF
50 SCREEN 3
60 COLOR 1,CF
70 '
80 X=100:Y=100                   ' coordonnee depart
90 '--------------- curseur clignotant
100 T=POINT(X,Y)
110 '
120 C$=INKEY$:IF C$<>"" THEN 170 ' test clavier
130 PSET(X,Y),CL
140 PSET(X,Y),CF
150 GOTO 120
160 '-------
170 IF L=0 THEN PSET(X,Y),CL
180 IF L=1 THEN PSET(X,Y),T
190 IF L=2 THEN PSET(X,Y),CF
200 '
210 C=ASC(C$)
220 IF C=29 THEN IF X>4 THEN X=X-4          ' gauche
230 IF C=28 THEN IF X<254 THEN X=X+4        ' droite
240 IF C=31 THEN IF Y<180 THEN Y=Y+4        ' bas
250 IF C=30 THEN IF Y>4 THEN Y=Y-4          ' haut
260 '
270 '-
280 IF C$="L" THEN L=1           ' lever
290 IF C$="B" THEN L=0           ' baisser
300 IF C$="E" THEN L=2           ' effacer
310 IF VAL(C$)>0 AND VAL(C$)<10 THEN CL=VAL(C$)  ' couleurs
320 GOTO 100
325 '
330 '  Pour diagonales,cf telecran haute resolution
```

LES SPRITES

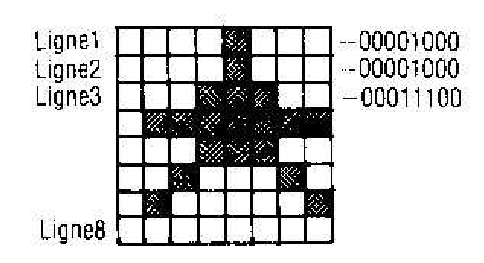
- SPRITE$()
- PUT SPRITE
- ON SPRITE GOSUB
- SPRITE ON
- SPRITE OFF
- SPRITE STOP

Les sprites (fantômes ou lutins) sont représentés par des matrices de 8×8 points ou 16×16 points.
Ils peuvent être déplacés rapidement et n'affectent pas le graphisme en place.
Ils sont acceptés dans les modes SCREEN1, SCREEN2 et SCREEN3.
La définition se fait avec **SPRITE$(n)** et l'affichage avec **PUT SPRITE**.

DEFINITION 8×8

Pour définir un sprite, on indique dans des lignes de DATA les points allumés(1) et les points éteints(0) puis on constitue une chaîne de caractères comme il est indiqué dans le programme.

Ci-dessous, nous avons représenté une étoile.




```
10 '--------- SPRITE 8X8
20 CL=4:COLOR CL,15        ' bleu sur fond blanc
30 SCREEN 2,0              ' 0:sprite 8*8 taille simple
40 DATA 00001000
50 DATA 00001000
60 DATA 00011100
70 DATA 01111111
80 DATA 00011100
90 DATA 00100010
100 DATA 01000001
110 DATA 00000000
120 '-----
130 FOR L=1 TO 8
140  READ L$
150  S$=S$+CHR$(VAL("&B"+L$))
160 NEXT L
170 '
180 SPRITE$(1)=S$
190 PUT SPRITE 1,(100,100),CL,1    ' sprite no1 dans plan no1
195 '
200 C$=INPUT$(1)
```

PUT SPRITE n° plan,STEP(X,Y),couleur,n° sprite

Les sprites peuvent etre affichés dans 32 "plans" différents (0 à 31).
Un sprite peut être affiché dans plusieurs plans mais un plan n'accepte qu'un sprite.
Les numéros de sprite doivent être compris entre 0 et 255 pour des sprites de 8×8, et entre 0 et 63 pour des sprites de 16×16.
Les coordonnées d'affichage spécifiées sont celles de la haute résolution (256×192).

PUT SPRITE2,(100,100),,1

Affiche le sprite 1 dans le plan 2 au point (100,100).
PUT SPRITE1,,,2 affiche le sprite 2 dans le plan 1 au point courant.
PUT SPRITE STEP(10,10),,2 déplace le sprite de (10,10).

TAILLE DES SPRITES ET ECHELLE

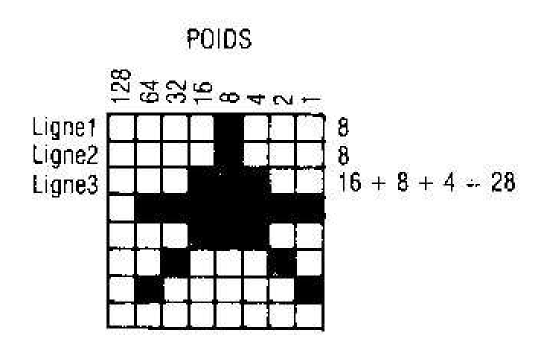
La taille des sprites (8×8 ou 16×16) ainsi que l'échelle (1 ou 2) sont définies par l'instruction "SCREEN mode, type sprite" :

0: 8×8 échelle 1
1: 8×8 échelle 2
2: 16×16 échelle 1
3: 16×16 échelle 2

Les coordonnées d'affichage X,Y doivent être comprises entre −32 et 255 pour X, −32 et 191 pour Y
Y=209 fait disparaître un sprite.
Y=208 fait disparaître tous les sprites des plans de numéro supérieurs au plan spécifié.
Si plusieurs sprites sont proches ou sur le même axe X, il faut limiter le nombre de sprites à 4.
SCREEN1 permet d'essayer PUT SPRITE en mode direct.

La définition en binaire des sprites est fastidieuse. On procède plutôt comme suit :
Pour chaque ligne de 8 points, on additionne les "poids" des points qui doivent être allumés. Sur l'exemple ci-dessous, pour la ligne 3, on obtient :

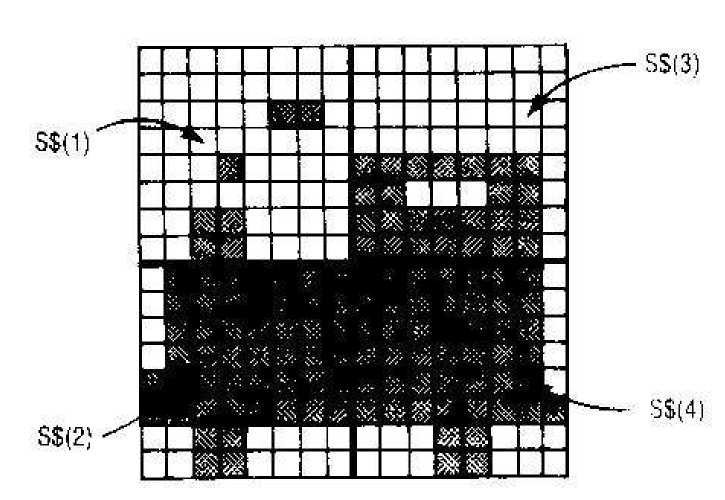
16 + 8 + 4 = 28.



```
10 '--------- SPRITE 8X8
20 CL=4:COLOR CL,15          ' bleu sur fond blanc
30 SCREEN 2,0                ' 0:sprite 8*8 taille simple
40 DATA 8,8,28,127,28,34,65,0
50 '-----
60 FOR L=1 TO 8
70  READ ND
80  S$=S$+CHR$(ND)
90 NEXT L
100 '
110 SPRITE$(1)=S$
120 PUT SPRITE 1,(100,100),CL,1    ' sprite no1 dans plan no1
130 '
140 C$=INPUT$(1)
```

DEFINITION 16×16

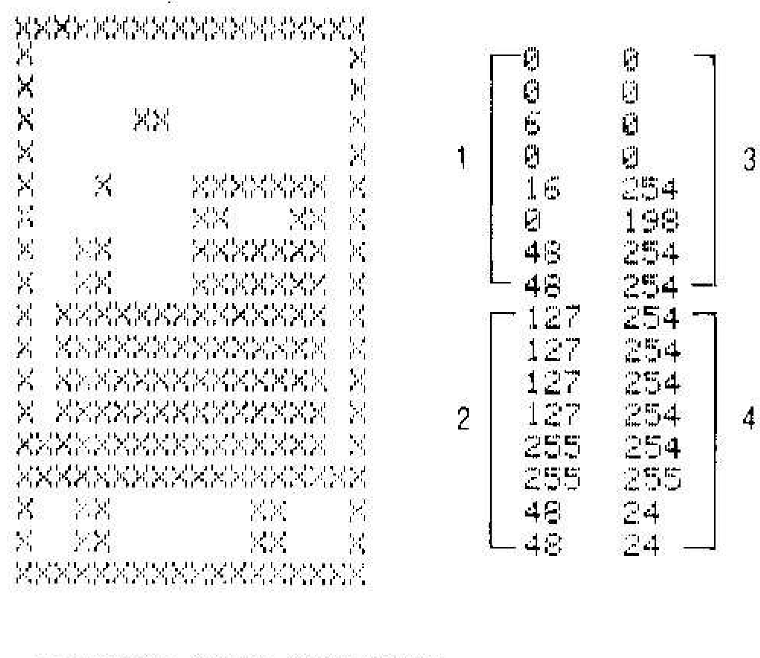
Pour représenter une figure de 16×16 points, il faut définir quatre sous-chaînes S$(1), S$(2), S$(3), S$(4) et les concaténer.




```
10 '------------ sprite 16x16
20 CLEAR 1000
30 SCREEN 2,2   '
35 '------------ lecture DATAS
40 FOR N=1 TO 4
50  FOR I=1 TO 8
60   READ X
70   S$(N)=S$(N)+CHR$(X)
80  NEXT I
90 NEXT N
100 SPRITE$(1)=S$(1)+S$(2)+S$(3)+S$(4)
110 PUT SPRITE 1,(100,100),1,1
120 '------------- dessin locomotive
130 DATA 0,0,6,0,16,0,48,48
140 DATA 127,127,127,127,255,255,48,48
150 DATA 0,0,0,0,254,198,254,254
160 DATA 254,254,254,254,255,255,24,24
170 '
180 C$=INPUT$(1)
```

■ Générateurs de sprites :

Nous proposons un programme générateur de sprites permettant de représenter des sprites de 16×16 points.
En mode "télécran", nous représentons le sprite. Le programme fournit ensuite les valeurs décimales.




```
FLECHES POUR DEPLACER
L:LEVER    B:BAISSER   F:FIN
```

Pour obtenir les valeurs de la figure retournée de 180°, ajouter les instructions suivantes :

```
450 LOCATE 15,19:INPUT "RETOURNEMENT(O/N)";R$:IF R$<>"O" THEN 150
630 '------------- RETOURNEMENT
640 FOR CL=1 TO 2                 ' 2 COLONNES
650  FOR L=1 TO 16                ' 16 LIGNES
660   PS=(CL-1)*8
670   ND=0                        ' VALEUR DECIMALE
680   FOR X=1 TO 8                ' 1 CARACTERE
690    A=0:IF VPEEK(AM+2+PS+X+L*40)=200 THEN A=1
700   ND=ND+A*2^(X-1)
710   NEXT X
720   LOCATE 20+((2-CL)+1 )*5,L:PRINT ND,SPC(1)
730  NEXT L
740 NEXT CL
750 GOTO 150
```

```
5 '--- GENERATEUR DE SPRITES (16*16)
10 SCREEN 0
20 COLOR 4,15
70 FOR X=0 TO 17:LOCATE X,0:PRINT "X":LOCATE X,17:
   PRINT "X":NEXT X
80 FOR Y=0 TO 17:LOCATE 0,Y:PRINT "X":LOCATE 17,Y:
   PRINT "X":NEXT Y
85 LOCATE 1,19:PRINT "CLAVIER MAJUSCULE"
90 LOCATE 1,20:PRINT "FLECHES POUR DEPLACER"
100 LOCATE 1,21:PRINT "L:LEVER  B:BAISSER  F:FIN"
110 X=10:Y=10                 ' COORDONNEES DEPART
140 '---------------- CURSEUR CLIGNOTANT
150 C$=INKEY$:IF C$<>"" THEN 200 'TEST CLAVIER
160 LOCATE X,Y:PRINT CHR$(200)
170 LOCATE X,Y:PRINT CHR$(32)
180 GOTO 150
190 '
200 IF L=0 THEN LOCATE X,Y:PRINT CHR$(200)
210 IF L=1 THEN LOCATE X,Y:PRINT CHR$(32)
220
230 C=ASC(C$)
240 IF C=29 THEN IF X>1 THEN X=X-1    'GAUCHE
250 IF C=28 THEN IF X<16 THEN X=X+1   'DROITE
260 IF C=31 THEN IF Y<16 THEN Y=Y+1   'BAS
270 IF C=30 THEN IF Y>1  THEN Y=Y-1   'HAUT
275 IF C=>32 THEN LOCATE 20,19:PRINT C$
280 IF C$="L" THEN L=1
290 IF C$="B" THEN L=0
300 IF C$="F" THEN 330
310 GOTO 150
320 '--------------------- CALCUL VALEURS DECIMALES
330 AM=BASE(0)
340 FOR CL=1 TO 2                  ' 2 COLONNES
350  FOR L=1 TO 16                 ' 16 LIGNES
360   PS=(CL-1)*8
370   ND=0                         ' VALEUR DECIMALE
380   FOR X=1 TO 8                 ' 1 CARACTERE
390    A=0:IF VPEEK(AM+2+PS+X+L*40)=200 THEN A=1
400   ND=ND+A*2^(8-X)
410   NEXT X
420   LOCATE 20+CL*5,L:PRINT ND;SPC(1);
430  NEXT L
440 NEXT CL
450 GOTO 150
```

ON SPRITE GOSUB n° ligne
SPRITE ON
SPRITE OFF
SPRITE STOP

L'instruction **ON SPRITE GOSUB** définit le numéro de ligne où il y aura branchement du programme s'il survient une collision de sprites.

SPRITE ON valide la déclaration faite par ON SPRITE GOSUB.
SPRITE OFF l'annule.
SPRITE STOP mémorise la collision en attendant SPRITE ON.

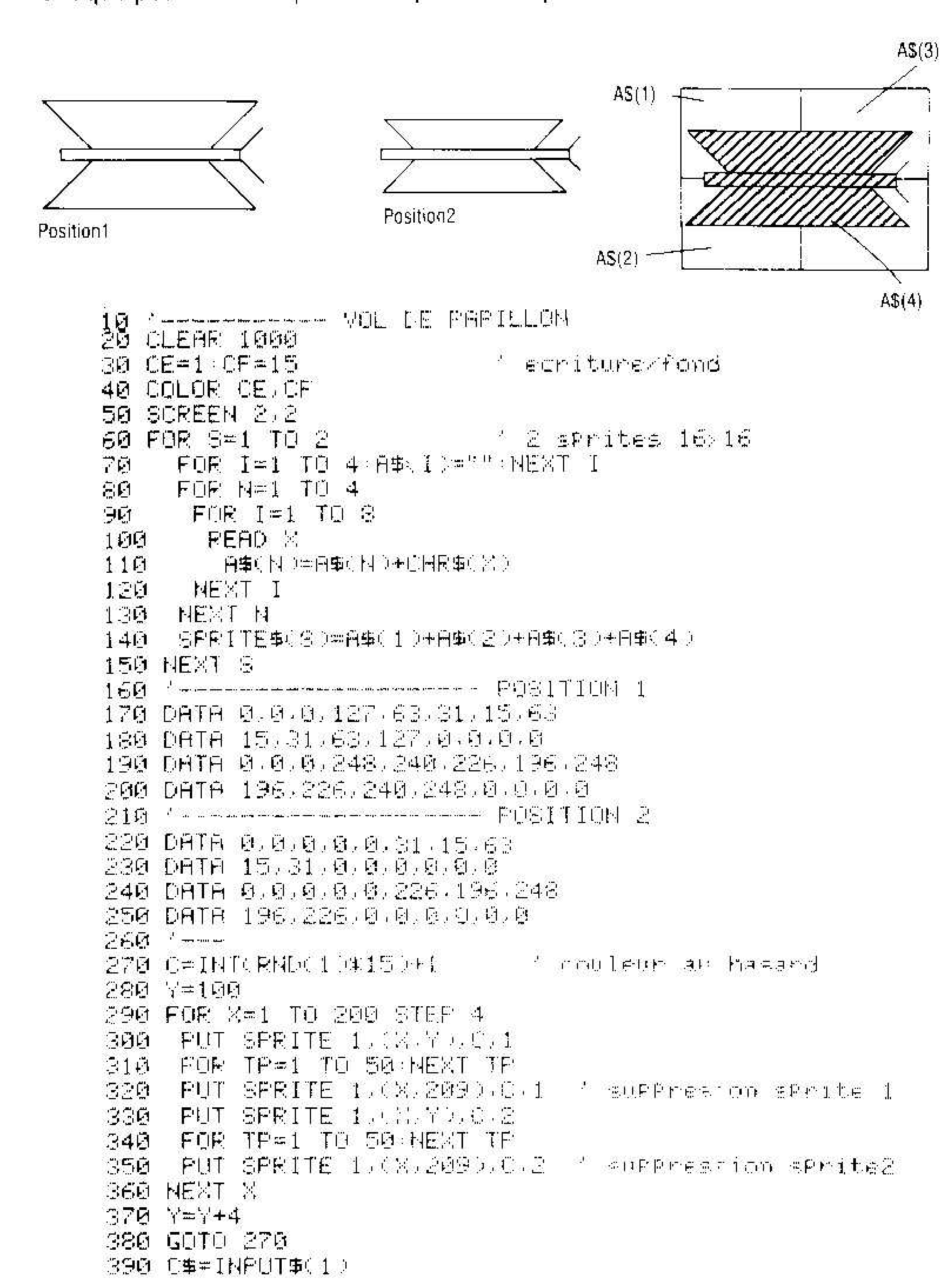
```
list
10 '--------------- collision sprite
20 COLOR 1,15:SCREEN 2,0
30 SPRITE$(1)=CHR$(8)+CHR$(8)+CHR$(28
)+CHR$(127)+CHR$(28)+CHR$(34)+CHR$(65
)
40 SPRITE$(2)=SPRITE$(1)
50 '
60 ON SPRITE GOSUB 140
70 SPRITE ON
80 Y=100
90 FOR X=1 TO 220
100  PUT SPRITE 1,(X,Y),1,1
110  PUT SPRITE 2,(220-X,Y),2,2
120 NEXT X
130 '------------------ collision
140 PLAY "CDE"
150 C$=INPUT$(1)
```

■ **Une étoile filante traverse l'écran :**

```
10 '---------- ETOILE FILANTE
20 CLEAR 1000
30 COLOR 1,15
40 SCREEN 2,2
50 FOR N=1 TO 4
60 FOR I=1 TO 8
70  READ X
80  A$(N)=A$(N)+CHR$(X)
90 NEXT I
100 NEXT N
110 SPRITE$(1)=A$(1)+A$(2)+A$(3)+A$(4)
120 '------- DEPLACEMENT
130 C=RND(1)*15
140 X=RND(1)*100+10:Y=1
150 S=INT(RND(1)*2):IF S=0 THEN S=-1
160 '
170  PUT SPRITE 1,(X,Y),C,1
180 X=X+S*3:Y=Y+3:IF X>190 OR X<10 THEN 130
190 GOTO 170
200 '
210 DATA 0,0,1,1,1,63,15,3
220 DATA 3,7,6,12,8,0,0,0
230 DATA 128,128,192,192,192,254,248,224
240 DATA 224,240,48,24,8,0,0,0
250 C$=INPUT$(1)
```

■ Vol de papillon

Pour simuler un vol de papillon, nous le représentons alternativement dans 2 positions.
Chaque position est représentée par 16×16 points.




```
10 '------------ VOL DE PAPILLON
20 CLEAR 1000
30 CE=1:CF=15                    ' ecriture/fond
40 COLOR CE,CF
50 SCREEN 2,2
60 FOR S=1 TO 2                  ' 2 sprites 16x16
70   FOR I=1 TO 4:A$(I)="":NEXT I
80   FOR N=1 TO 4
90    FOR I=1 TO 8
100     READ X
110      A$(N)=A$(N)+CHR$(X)
120    NEXT I
130   NEXT N
140   SPRITE$(S)=A$(1)+A$(2)+A$(3)+A$(4)
150 NEXT S
160 '------------------------- POSITION 1
170 DATA 0,0,0,127,63,31,15,63
180 DATA 15,31,63,127,0,0,0,0
190 DATA 0,0,0,248,240,226,196,248
200 DATA 196,226,240,248,0,0,0,0
210 '------------------------- POSITION 2
220 DATA 0,0,0,0,0,31,15,63
230 DATA 15,31,0,0,0,0,0,0
240 DATA 0,0,0,0,0,226,196,248
250 DATA 196,226,0,0,0,0,0,0
260 '---
270 C=INT(RND(1)*15)+1           ' couleur au hasard
280 Y=100
290 FOR X=1 TO 200 STEP 4
300  PUT SPRITE 1,(X,Y),C,1
310  FOR TP=1 TO 50:NEXT TP
320  PUT SPRITE 1,(X,209),C,1   ' suppression sprite 1
330  PUT SPRITE 1,(X,Y),C,2
340  FOR TP=1 TO 50:NEXT TP
350  PUT SPRITE 1,(X,209),C,2   ' suppression sprite2
360 NEXT X
370 Y=Y+4
380 GOTO 270
390 C$=INPUT$(1)
```

■ **Une étoile est affichée dans 32 plans.**

```
10 '------------- ciel etoile multicolore
20 CLEAR 1000
30 COLOR 1,15,15
40 SCREEN 2,2
50 FOR N=1 TO 4
60  FOR I=1 TO 8
70   READ X
80   A$(N)=A$(N)+CHR$(X)
90  NEXT I
100 NEXT N
110 '
120 SPRITE$(1)=A$(1)+A$(2)+A$(3)+A$(4)
130 '---------------- affichage 32 sprites
140 FOR P=0 TO 31
150  X=RND(1)*220+10:Y=RND(1)*170+10
160  CL=RND(1)*15+1
170  PUT SPRITE P,(X,Y),CL,1
180 NEXT P
190 C$=INPUT$(1)
200 '--------------- dessin etoile
210 DATA 0,0,1,1,1,63,15,3
220 DATA 3,7,6,12,8,0,0,0
230 DATA 128,128,192,192,192,254,248,224
240 DATA 224,240,48,24,8,0,0,0
```

■ Ci-dessous, nous représentons un **soleil en haute résolution** avec un cercle plein et huit rayons. La deuxième partie du programme fournit les valeurs décimales pour représenter le soleil sous forme d'un sprite.

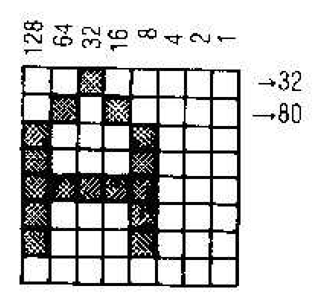
```
60 '--------- dessin de soleil
70 CF=15                          ' couleur de fond
80 XC=8:YC=8:R=3                  ' centre/rayon
90 COLOR 1,CF
100 SCREEN 2                      ' haute resolution
110 CIRCLE(XC,YC),R,1
120 PAINT(XC,YC),1
130 FOR A=0 TO 2*3.14 STEP 3.15/4   ' rayon
140  R1=R*2.5
150  X=XC+R1*COS(A)
160  Y=YC+R1*SIN(A)
170  LINE (XC,YC)-(X,Y)
180 NEXT A
190 '
200 '----------- CALCUL VALEURS DECIMALES
210 OPEN "GRP:" FOR OUTPUT AS #1
220 FOR CL=1 TO 2
230  FOR L=1 TO 16
240   PS=(CL-1)*8
250   ND=0
260   FOR X=1 TO 8
270    A=0:IF POINT(PS+X,L)<>CF THEN A=1
280    ND=ND+A*2^(8-X)
```

```
290    NEXT X
300    X1=100+CL*32:Y1=40+L*8
310    DRAW "BM=X1;,=Y1;"
320    PRINT #1,ND
330 NEXT L
340 NEXT CL
350 C$=INPUT$(1)
```

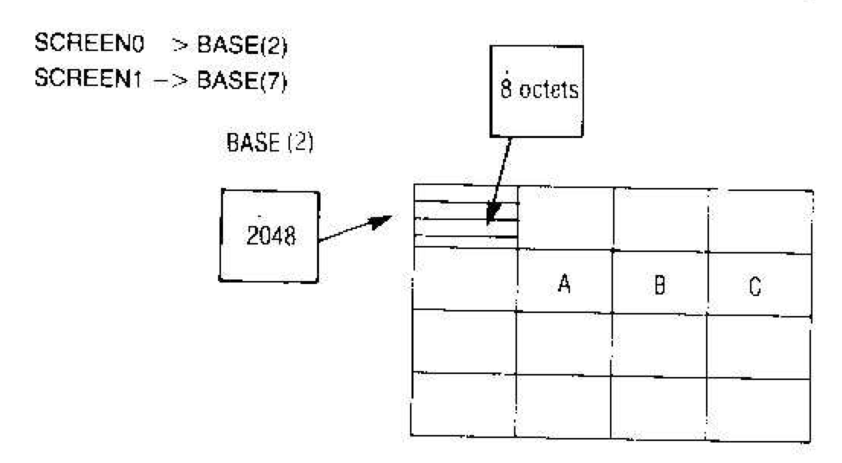
```
10 '------------- dessin soleil obtenu
20 DATA 1,65,33,17,11,7,239,31
30 DATA 15,7,27,34,66,2,2,0
40 DATA 0,8,16,32,192,192,224,254
50 DATA 224,192,160,16,8,0,0,0
```

REDÉFINITION DE CARACTÈRES

Chaque caractère est représenté en mémoire sous forme de 8 octets.

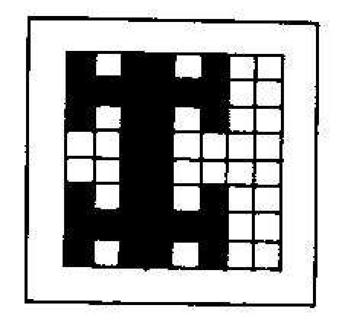


L'adresse du "générateur de caractères" est donnée par l'instruction BASE(n).



L'adresse du premier octet d'un caractère est donnée par :
adresse=BASE(2)+ASC(caractère)*8.
Pour modifier un caractère, on utilise VPOKE adresse, valeur.
En mode "SCREEN0", seuls les 6 bits de gauche des caractères apparaissent à l'écran.

■ Ci-dessous, le caractère "**C**" est remplacé par un dessin de voiture.

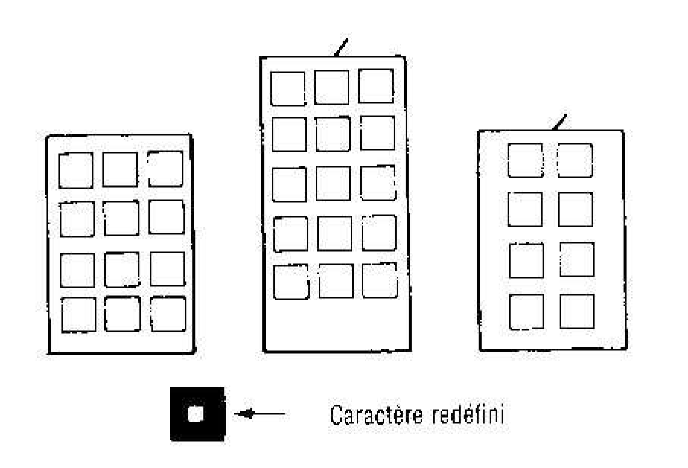


```
10 '--------- REDEFINITION CARACTERES
20 '
30 '---- dessin voiture
40 '
50 DATA 180,252,180,48,48,180,252,180
60 '
70 SCREEN 0
80 C$="C"                    ' caractere a modifier
90 AG=BASE(2)                ' adresse generateur de caracteres
100 AC=AG+ASC(C$)*8
110 '
120 FOR M=AC TO AC+7
130  READ ND
140  VPOKE M,ND
150 NEXT M
160 PRINT C$
170 STOP
```

■ Le programme ci-dessous inverse le caractère "**A**" :

```
180 '-------------- INVERSION DE  A'
190 '
200 SCREEN 0
210 C$="A"
220 AG=BASE(2)
230 AC=AG+ASC(C$)*8
240 '
250 FOR I=0 TO 7
260  M=AC+I
270  C(I)=VPEEK(M)
280 NEXT I
290 '
300 FOR I=0 TO 7
310  M=AC+I
320  VPOKE M,C(7-I)
330 NEXT I
340 PRINT C$
```

■ Après avoir redéfini le caractère "**C**", nous dessinons des immeubles :



```
10 '------------------ DESSIN IMMEUBLE
20 DATA 252,132,132,132,132,132,132,252
30 '
40 SCREEN 0
50 C$="C"                  ' caractere a modifier
60 AG=BASE(2)              ' adresse generateur de caracteres
70 AC=AG+ASC(C$)*8
80 '
90 FOR M=AC TO AC+7
100  READ ND
110  VPOKE M,ND
120 NEXT M
130 '----
140 XB=1:YB=20
150 '--
160 FOR N=1 TO 5                  ' 5 immeubles
170 H=RND(1)*5+5:L=RND(1)*3+3
180  FOR Y=YB TO YB-H STEP -1
190   FOR X=XB TO XB+L
200   LOCATE X,Y:PRINT C$;
210   NEXT X
220  NEXT Y
230 XB=XB+L+2
240 NEXT N
```

■ Le programme ci-dessous représente une maison utilisant quatre caractères (A B C D).
CHR$(8) déplace le curseur à gauche et CHR$(31) vers le bas.

```
10 '------ DESSIN MAISON (MODE SCREEN 1)
20 '
30 '-------- 4 CARACTERES
40 '
50 DATA 12,12,63,127,255,64,95,85
60 DATA 0,0,248,252,255,2,2,2
70 DATA 95,85,85,85,95,64,64,127
80 DATA 122,74,74,106,74,74,74,254
90 '
100 SCREEN 1
110 C$="A"               ' 1er caractere a modifier
120 AG=BASE(7)           ' adresse generateur de caracteres
130 AC=AG+ASC(C$)*8
140 '
150 FOR N=1 TO 4
160  A=AC+(N-1)*8
170  FOR M=A TO A+7
180   READ ND
190   VPOKE M,ND
200  NEXT M
210 NEXT N
220 '--
230 C=ASC(C$)
240 K$=CHR$(C)+CHR$(C+1)+CHR$(31)+CHR$(8)+CHR$(8)+CHR$(C+2)+CHR$(C+3)
250 '------ AFFICHAGE
260 FOR N=1 TO 10
270  X=RND(1)*30:Y=RND(1)*20
280  LOCATE X,Y:PRINT K$
290 NEXT N
```

MODIFICATION ECRAN EN MODE "SCREEN2"

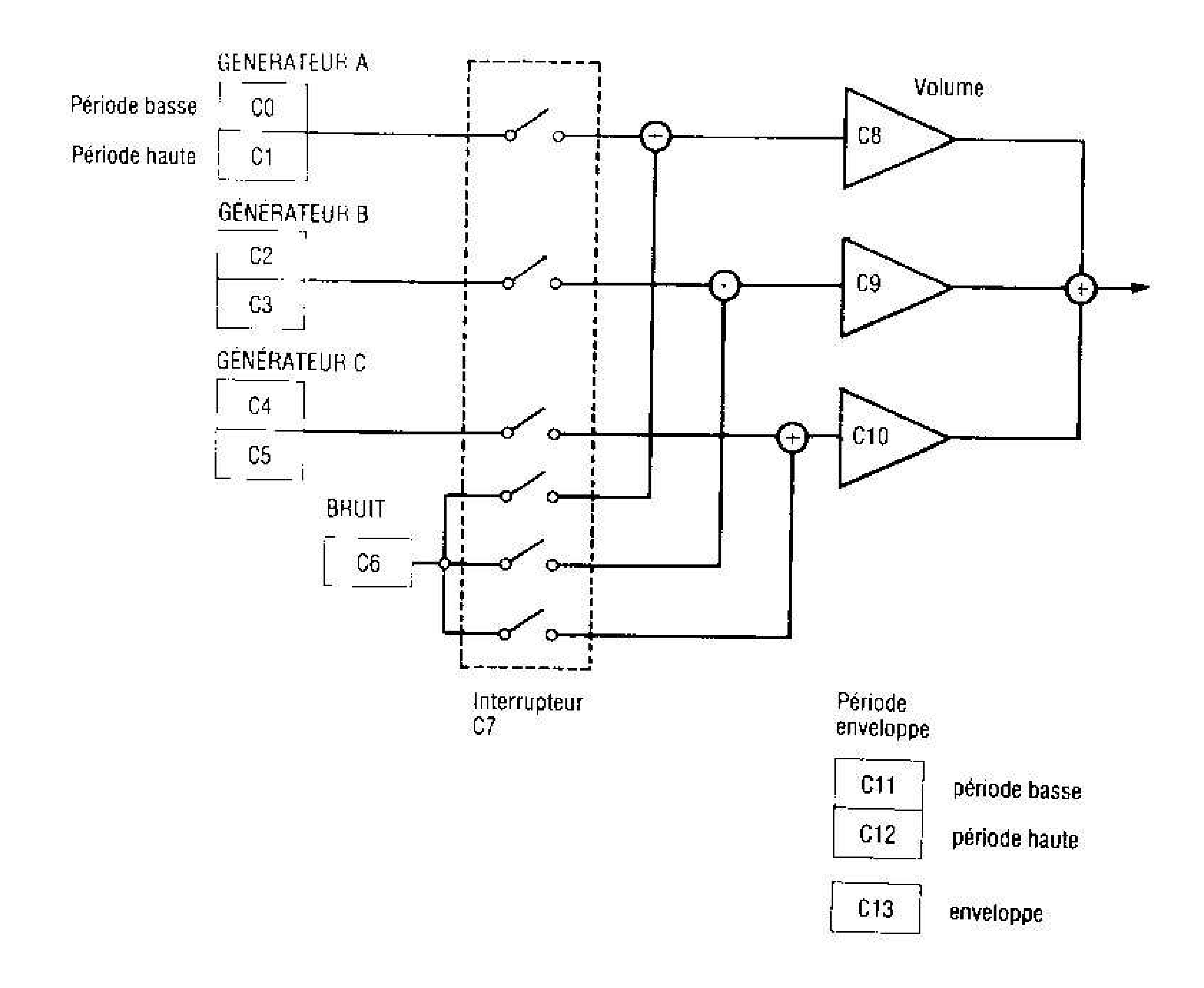
■ Ci-dessous, nous dessinons deux voitures en haute résolution en utilisant VPOKE.

```
10 '------------- Affichage direct ecran (SCREEN2)
20 '
30 '--------- dessin voiture
40 DATA 180,252,180,48,48,180,252,180
50 SCREEN 2
60 FOR I=0 TO 7:READ C(I):NEXT I
70 '
80 COL=10:LIG=2:GOSUB 130
90 LIG=3:COL=9:GOSUB 130
100 '
110 C$=INPUT$(1)
120 '-------------- sPgm 1 caractere
130 M=(LIG-1)*32*8+(COL-1)*8
140 FOR I=0 TO 7
150  VPOKE M+I,C(I)
160 NEXT I
170 RETURN
```

LES SONS

- ■ BEEP ■ PLAY
- ■ SOUND ■ PLAY(n)

Les sons sont fournis par trois générateurs (A,B,C,). Un bruit peut être ajouté à chacun de ces générateurs.
Plusieurs types d'enveloppes sont disponibles (8). 14 "canaux" permettent de commander la fréquence des générateurs (C0 à C5), le bruit (C6), le volume (C8 à C9) et l'enveloppe (C11 à C13).



BEEP

Provoque un son bref.

SOUND n° canal,valeur

Commande les différents canaux :

■ **Générateurs de fréquence A,B,C :**

Chaque générateur est commandé par deux registres de 8 bits dans lesquels on place la "période basse" et la "période haute".
La période réelle est donnée par :

Période=période haute*256+période basse

La fréquence est égale à : 124000/période.

Exemple : pour le générateur A (C0,C1), plaçons 100 dans C1 et 1 dans C0 :

```
SOUND 0,1
SOUND 1,100 -> 1*256+100=356
```

La fréquence est 124000/356.
La partie haute de la période doit être inférieure à 16 (4 bits seulement sont utilisés).

■ **Bruit (C6)**

Le canal C6 commande le bruit qui peut être ajouté aux fréquences A,B,C.

■ **Interrupteur (C7)**

Le canal C7 sélectionne les générateurs et le bruit. Un "zéro" commande le passage.

SOUND 7,&B111000	sélectionne A,B,C
SOUND 7,&B111110	sélectionne A seulement
SOUND 7,255-1	sélectionne A seulement
SOUND 7,0	sélectionne A,B,C et bruit pour A,B,C.

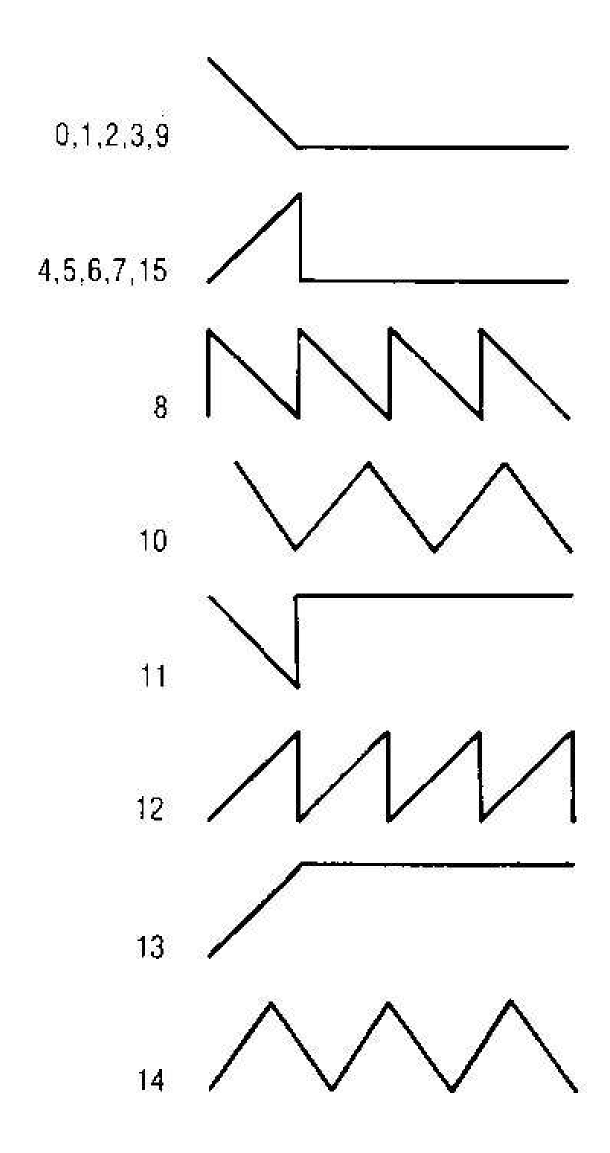
■ **Volume :**

Le volume doit être compris entre 0 et 15.
SOUND 8,15 règle le volume pour A au maximum.
"volume" égal à 16 indique que la fréquence doit être modulée par le générateur d'enveloppe.
Le programme ci-dessous vous permet de tester les notions ci-dessus.

```
10 '--------------- essai frequence
20 INPUT "FREQUENCE ";F
30 IF F<33 THEN 20
40 '
50 P=1/F*124000!              ' periode
60 C1=P\256                   ' canal 1
70 C0=P MOD 256               ' canal 0
80 PRINT C1,C0
90 '
100 SOUND 7,254               ' voix A
110 SOUND 8,12                ' volume 12 pour A
120 SOUND 0,C0
130 SOUND 1,C1
140 GOTO 20
```

■ Forme enveloppe (C13) et période enveloppe (C11 et C12) :

Huit formes d'enveloppe sont disponibles.
La forme se choisit par : SOUND 13,N° FORME.



La période de l'enveloppe est choisie par :

SOUND 11,période basse
SOUND 12,période haute

La période réelle est donnée par :

P=période haute*256+période basse

La fréquence est égale à : 7800/P.

Ce programme vous permet de tester les notions présentées ci-dessus.

```
10 '---------- essai frequence enveloppe
20 ' bruit equivalent a PING
30 '
40 INPUT "FREQUENCE enveloppe ";FE
50 '
60 '
70 P=1/FE*7800              ' periode
80 P2=P\256                 ' canal 12
90 P1=P MOD 256             ' canal 11
100 PRINT P2,P1
110 '
120 SOUND 7,254             ' voix A
130 SOUND 8,16              ' controle par enveloppe
140 SOUND 0,50:SOUND 1,1    ' frequence fixe
150 SOUND 13,8              ' type enveloppe
160 SOUND 11,P1             ' periode basse
170 SOUND 12,P2             ' periode haute
180 GOTO 10
```

Nous présentons quelques programmes de sons.

■ **Ping :**

```
10 '------------------------- ping
20 SOUND 7,254             ' voix A
30 SOUND 8,16              ' controle par enveloppe
40 SOUND 0,50:SOUND 1,1    ' frequence fixe
50 SOUND 13,8              ' type enveloppe
60 SOUND 11,0:SOUND 12,10
80 FOR TP=1 TO 100:NEXT TP
90 SOUND 8,0
```

■ **Explode :**

```
5 '---------------- explode
10 SOUND 6,20
20 SOUND 7,7:SOUND 12,26
30 FOR C=8 TO 10:SOUND C,16:NEXT C
40 SOUND 13,0
```

■ Shoot

```
5 '---------------- shoot --
10 SOUND 6,11
20 SOUND 7,7:SOUND 12,12
30 FOR C=8 TO 10:SOUND C,16:NEXT C
40 SOUND 13,0
```

■ Sirène

```
10 '------------- sirene
20 SOUND 7,254      ' generateur A
30 SOUND 8,10       ' volume pour A
40 '
50 FOR I=1 TO 255
60  SOUND 0,I
70 NEXT I
80 GOTO 50
```

■ Vague

```
5 '---------------- vague --
10 SOUND 6,11
20 SOUND 7,7:SOUND 12,40
30 FOR C=8 TO 10:SOUND C,16:NEXT C
40 SOUND 13,14
```

■ Descente soucoupe

```
10 '-------------- descente soucoupe
20 COLOR 1,15:SCREEN 2,0
30 SPRITE$(1)=CHR$(48)+CHR$(48)+CHR$(
48)+CHR$(120)+CHR$(252)+CHR$(252)+CHR
$(252)+CHR$(252)
40 '
50 Y=50
60 SOUND 7,254:SOUND 8,12   ' generate
ur A/volume=12
70 '
80 S=1                      ' sens
90 X1=10:X2=200             ' bornes
100 CL=RND(1)*14+1          ' couleur
110 FOR X=X1 TO X2 STEP S   ' avance
120  SOUND 0,X/2
130  PUT SPRITE 1,(X,Y),CL,1
140 NEXT X
150 S=-S:SWAP X1,X2         ' inversio
n sens
160 Y=Y+8                  ' descente
170 GOTO 100
```

PLAY chaîne1,chaîne2,chaîne3

Joue des notes sur trois voies simultanément. Les notes sont représentées par A,B,...F,G.

A:LA
B:SI
C:DO
D:RE
E:MI
F:FA
G:SOL

PLAY "CDEFGAB" joue DO,RE,...LA,SI

Les chaînes ne doivent pas comporter plus de 255 caractères. Une note peut être suivie par # ou + (diese) ou par − (bemol).

PLAY "C+D+E+"

Plusieurs commandes sont disponibles. Leur effet dure jusqu'à ce qu'une nouvelle commande définisse une nouvelle valeur. Par conséquent, on pensera à initialiser des valeurs qui auraient pu être changées par un programme précédent.
BEEP initialise ces valeurs.

■ **Octave : On :**

Change le niveau d'octave pour toutes les notes qui suivent la commande.
"n" doit être compris entre 1 et 8. La valeur par défaut est 4.

```
PLAY "O5CDEO4CDE"
```

■ **Note : Nn :**

Joue une note comprise entre 1 et 96 (8 octaves et demi ton).
PLAY "N36" est équivalent à **PLAY "O4C"**

```
20 PLAY "O4CDE"          ' DO RE MI
30 PLAY "N36N38N40"      ' c'est la meme chose
```

n=0 provoque un silence.

■ **Longueur : Ln :**

Détermine la longueur des notes après la commande. 1 donne la durée la plus longue et 64 la plus courte. Par défaut, "n" est égal à 4. Pour modifier la durée de la note suivante seulement, spécifier la valeur après la note.

```
10 PLAY "L32CDEFGAB"      ' toutes les notes
15 PLAY "RRR"
20 PLAY "L4CD32EFGAB"     ' D seulement
```

Chaque "." après une note multiplie sa longueur par 3/2 (2 points multiplient par 9/4).

```
10 PLAY "A..BCDFG"
```

■ **Silence : Rn :**

Provoque un silence. "n" doit être compris entre 1 et 64.

■ **Tempo : Tn :**

Détermine le tempo (32 à 255). La valeur par défaut est 120.

■ **Volume : Vn :**

Détermine le volume (0 à 15) qui est par défaut égal à 8.

```
10 PLAY "V2AV4AV8AV15A"
```

■ **Forme enveloppe : Sn :**

La forme de l'enveloppe doit être comprise entre 0 et 15 (cf. SOUND).

■ **Période enveloppe : Mn :**

Définit la période de l'enveloppe (1 à 65535).
Par défaut, n=255.

```
10 PLAY "S13M1000ABCDEFG"
```

■ **Passage de variables chaînes : Xvar$;**

Le passage d'une variable chaîne se fait en ajoutant "X" devant le nom de la variable et point-virgule après.

```
10 C1$="CDE"
20 C2$="FGA"
30 PLAY."XC1$;XC2$;"
```

■ **Passage de variables numériques : =var ;**

Pour spécifier une valeur donnée par une variable numérique, on place le signe "=" devant le nom de celle-ci et point-virgule après.
Le programme ci-dessous joue les notes de 36 à 96.

```
10 CLS
20 PLAY "L2"
30 FOR NT=36TO 96
40  LOCATE 10,10:PRINT NT
50  PLAY "N=NT;
60 NEXT NT
```

Exemples divers :

```
10 PLAY "C","E","G"          ' accord DO MI SOL

10 PLAY "T200ABCDEF"         ' tempo rapide
20 PLAY "T32ABCDEF"          ' tempo lent
```

```
5 '------------------------ musique aleatoire
10 DUR=3+INT(RND(1)*10)   ' duree au hasard
20 NT=INT(RND(1)*20)+32   ' note au hasard
30 '
40 PLAY "L=DUR;N=NT;"
50 GOTO 10
```

PLAY(n)

PLAY **chaîne1,chaîne2,chaîne3,** ne bloque pas le programme pendant que la musique est jouée.
L'instruction suivant PLAY est exécutée immédiatement.
PLAY(N) teste si une voie est libre (n=1,2,3)
PLAY(O) teste si toutes les voies sont libres.
Le programme ci-dessous attend que la musique soit jouée avant de poursuivre l'exécution.

```
10 A$="BCDEFG"
40 PLAY A$,A$,A$
50 IF PLAY(0)=-1 THEN 50     ' boucle d'attente
60 PRINT "suite"
```

Celui-ci affiche l'heure pendant la durée de la musique.

```
10 CLS
15 TIME=0
20 A$="BCDEFG"
30 PLAY A$,A$,A$
40 IF PLAY(0)=-1 THEN GOSUB 70:GOTO 40
50 GOTO 40
60 '--------------- horloge
70 LOCATE 10,10:PRINT TIME/50:RETURN
```

BEEP stoppe une musique en cours.

```
10 CLS
20 A$="ABCDEFG"
30 PLAY A$+A$+A$
40 '
50 IF INKEY$<>"" THEN BEEP
60 GOTO 50
```

■ Morse :

Ce programme vous permet d'apprendre l'alphabet MORSE. Par exemple, en appuyant sur la lettre S, vous entendez 3 sons brefs. En appuyant sur 0, vous entendrez 3 sons longs.

```
10 '---------- MORSE
20 '
30 CLEAR 1000
40 DIM MR$(26)
50 DATA .-,-...,-.-.
60 DATA -..,.,..-.
70 DATA --.,....,..,.---,-.-,.-..,--
80 DATA -.,---,.--.,--.-,.-.,...,-
90 DATA ..-,...-,.--,-..-,-.--,--..
100 CLS
110 FOR I=1 TO 26:READ MR$(I):NEXT
120 '
130 C$=INPUT$(1)
140 '
150 P=ASC(C$)-64:IF P<1 OR P>26 THEN 130
160 '
170 X$=MR$(P)
175 PRINT MR$(P);" ";
180 FOR P=1 TO LEN(X$)
190  IF MID$(X$,P,1)="." THEN PLAY"L12A"
200  IF MID$(X$,P,1)="-" THEN PLAY"L5A"
210  FOR TP=1 TO 20:NEXT TP
220 NEXT P
230 GOTO 130
235 '==============================
250 '  POUR AFFICHER EN MORSE:
260 '   175 PRINT MR$(P);" ";
```

A	• —
B	— •••
C	— • — •
Z	— — ••

Table MRS$( )

■ Pour jouer avec le clavier :

A chaque touche du clavier, nous faisons correspondre une note.

```
10 '---------- MUSIQUE CLAVIER     suppression sono clavier
20 SCREEN ,,0
30 DIM CL$(40),NTE$(40)
40 '
50 DATA Q,O3A,W,O3B,E,O3C,R,O3D
60 DATA A,O4A,S,O4B,D,O4C,F,O4D
70 DATA FIN
80 '
90 FOR N=1 TO 40
100  READ X$:IF X$="FIN" THEN 150
110 READ Y$
120 CL$(N)=X$:NTE$(N)=Y$
130 NEXT N
140 '----------------
150 X$=INPUT$(1)
160 FOR P=1 TO N
170  IF CL$(P)=X$ THEN PLAY NTE$(P):GOTO 150
180 NEXT P
190 GOTO 150
```

CL$( )		NTE$ ( )
Q	↔	03A
W	↔	03B
E	↔	03C
A		
S		
D		

Q	W	E
A	S	D

LES FICHIERS SÉQUENTIELS | 10

- ■ OPEN
- ■ PRINT#
- ■ INPUT#
- ■ EOF
- ■ CLOSE
- ■ INPUT$#
- ■ LINE INPUT#
- ■ MOTOR
- ■ MOTOR ON
- ■ MOTOR OFF
- ■ MAXFILES

DUPONT	RC	LF	955-19-07	RC	LF	

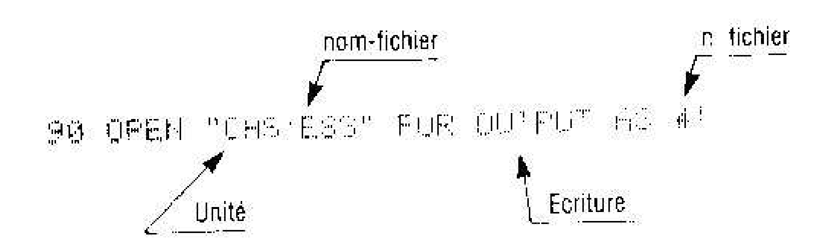
Le stockage d'informations s'y fait en fin de fichier au fur et à mesure des arrivées. Une information particulière ne pouvant y être ensuite retrouvée qu'après avoir lu toutes les précédentes (accès séquentiel), leur organisation est dite séquentielle. Les instructions d'écriture et de lecture sont analogues à celles de l'impression et de la lecture au clavier. (PRINT – INPUT).
Dans le cas le plus simple, les enregistrements sont séparés, de façon interne et donc transparente à l'utilisateur, par des caractères "Carriage Return" (Retour chariot) et "Line Feed" (saut de ligne) (Codes ASCII 13 et 10).

OPEN "U:nom-fichier" FOR mode AS #n° fichier

Un fichier est ouvert en écriture (OUTPUT), en lecture (INPUT) ou en ajout (APPEND).

OUTPUT : écriture
INPUT : lecture
APPEND : ajout en fin de fichier (disque seulement).

Le lecteur de cassette est spécifié par "CAS :". S'il n'existe pas de lecteur de disque, "CAS :" peut être omis.



"OPEN" réserve une mémoire tampon et lui affecte un numéro (# 1 sur exemple). C'est ce numéro qui est utilisé dans les ordres d'écriture et de lecture.

Lors des opérations de lecture et d'écriture, les informations transitent par la mémoire tampon. Pour une écriture, c'est seulement lorsque la mémoire tampon est pleine que le transfert vers la cassette (ou la disquette) a lieu.

Remarque :

OPEN : sert également à ouvrir des "canaux" vers des périphériques tels que :
CRT : écran SCREEN 0 ou 1
GRP : écran SCREEN 2
LPT : imprimante

(cf. chapitre "Les éditions").

PRINT#, n° fichier, variable
INPUT#, n° fichier, variable

Dans le cas le plus simple, l'écriture dans un fichier séquentiel se fait par :

PRINT # n° fichier, variable

Si l'utilisateur programme :

```
110 INPUT "NOM (OU FIN) ";NOM$
120 INPUT "TELEPHONE ";TPH$
130 IF NOM$="FIN" THEN CLOSE#1 GOTO 190
140 PRINT #1,NOM$
150 PRINT #1,TPH$
```

il y a écriture de 2 enregistrements qui seront lus par :

INPUT # n° fichier, variable, variable

```
270 INPUT #1,NOM$,TPH$
280 PRINT NOM$,TPH$
```

(Les 2 enregistrements peuvent également être lus par 2 instructions INPUT#).

Les lectures doivent, bien sûr, s'effectuer dans l'ordre où les écritures ont été faites.

Attention :

```
PRINT #1,NOM$
PRINT #1,TPH$
```

ne doit pas être remplacé par :

```
PRINT #1,NOM$,TPH$
```

Les variables seraient séparées par des espaces seulement (comme pour PRINT NOM$, TPH$ à l'écran).

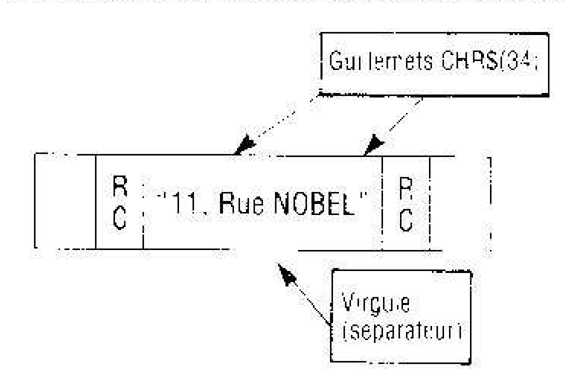
CHAINE COMPORTANT DES VIRGULES

La virgule étant considérée comme séparateur, il faut, lors d'une écriture par PRINT #, encadrer la chaîne par des guillemets (CHR$(34)). Elle peut ainsi être relue par INPUT #.

Si les guillemets n'ont pas été prévus à l'écriture, il faut utiliser LINE INPUT # à la lecture.



```
PRINT#1,CHR$(34);RUE$;CHR$(34)
```



EOF(n° fichier)

Cette instruction repère la fin d'un fichier en lecture. Elle doit être placée AVANT l'instruction de lecture INPUT #.

```
260 IF EOF(1)=-1 THEN CLOSE #1:GOTO 320
270 INPUT #1,NOM$,TPH$
```

La fin des fichiers est repérée par le caractère CHR$(26).

CLOSE # n° de fichier

Transfère le contenu de la mémoire tampon dans le fichier puis libère celle-ci. "CLOSE" sans numéro de fichier ferme tous les fichiers.

```
CLOSE #1
```

Cette instruction est obligatoire.
RUN, NEW et END ferment les fichiers mais pas STOP.

INPUT$(N,# n° fichier)

Lit N caractères d'un fichier.
Les séparateurs sont traités comme les autres caractères.

```
410 L$=INPUT$(1,#1)
```

LINE INPUT#n° fichier, chaîne

Lit une chaîne de caractères (255 maximum) en ne considérant comme séparateur que le retour chariot (code 13).
La virgule n'est pas considérée comme séparateur.

```
500 LINE INPUT #1,LIG$
```

MOTOR

Met en marche le moteur du lecteur de cassette et le stoppe (bascule) si la télécommande est connectée.

MOTOR ON

Met en marche le moteur du lecteur de cassette.

MOTOR OFF

Stoppe le moteur du lecteur de cassette.

MAXFILES=nombre

Définit le nombre maximum de fichiers ouverts simultanément (1 par défaut).
MAXFILES=0 ne permet plus que la sauvegarde des programmes.
MAXFILES réserve une mémoire tampon de 267 octets par fichier.
MAXFILES remet les variables à zéro. Par conséquent, elle doit être écrite en début de programme.

```
10 X=123
20 MAXFILES=2
30 PRINT X

RUN
0
```

Un édition peut être provisoirement stockée sur cassette en ajoutant un numéro de fichier devant les instructions PRINT. L'édition des résultats se fait en lisant le fichier.

Le programme ci-dessous, illustrant les instructions des fichiers séquentiels ne fonctionne qu'avec la télécommande, puisque l'écriture sur cassette ne se fait pas de façon continue.

```
10 '-------ESSAI FICHIER SEQUENTIEL  CASSETTE (AVEC TELECOMMANDE)
20 '
30 '------------ ECRITURE
40 PRINT "PLACEZ LA TELECOMMANDE"
50 PRINT "APPUYER SUR <RECORD> DU CASSETTE"
60 PRINT "PUIS <RETURN>"
70 C$=INPUT$(1)
80 '
90 OPEN "CAS:ESS" FOR OUTPUT AS #1
100 '
110 INPUT "NOM (OU FIN) ";NOM$
120 INPUT "TELEPHONE ";TPH$
130 IF NOM$="FIN" THEN CLOSE#1:GOTO 190
140 PRINT #1,NOM$
150 PRINT #1,TPH$
160 GOTO 110
170 '-------------- LECTURE AVEC INPUT#
180 '
190 PRINT:PRINT "REMBOBINEZ(SANS TELECOMMANDE) PUIS"
200 PRINT "APPUYEZ SUR <RETURN> "
210 PRINT "PUIS <PLAY> (AVEC TELECOMMANDE)"
220 C$=INPUT$(1)
230 '
```

→

```
240 OPEN "CAS:ESS" FOR INPUT AS #1
250 '
260 IF EOF(1)=-1 THEN CLOSE #1:GOTO 320
270 INPUT #1,NOM$,TPH$
280 PRINT NOM$,TPH$
290 GOTO 260
300 '------------------ LECTURE AVEC INPUT$(1,#1)
310 '
320 PRINT:PRINT "REMBOBINEZ(SANS TELECOMMANDE) PUIS"
330 PRINT "APPUYEZ SUR <RETURN> "
340 PRINT "PUIS <PLAY> (AVEC TELECOMMANDE)"
350 C$=INPUT$(1)
360 '
370 OPEN "CAS:ESS" FOR INPUT AS #1
380 '
390 IF EOF(1)=-1 THEN CLOSE #1:END
400 '
410 C$=INPUT$(1,#1)
420 PRINT C$,ASC(C$)
430 GOTO 390
```

```
PLACEZ LA TELECOMMANDE
APPUYER SUR <RECORD> DU CASSETTE
PUIS <RETURN>
NOM (OU FIN) ? BALU
TELEPHONE ? 1111
NOM (OU FIN) ? LABU
TELEPHONE ? 2222
NOM (OU FIN) ? FIN
TELEPHONE ? 3333

REMBOBINEZ(SANS TELECOMMANDE) PUIS
APPUYEZ SUR <RETURN>
PUIS <PLAY> (AVEC TELECOMMANDE)
BALU          1111
LABU          2222

REMBOBINEZ(SANS TELECOMMANDE) PUIS
APPUYEZ SUR <RETURN>
PUIS <PLAY> (AVEC TELECOMMANDE)


RUN

B              66
A              65
L              76
U              85
               13

               10
1              49
1              49
1              49
1              49
               13

               10
L              76
A              6
```

■ Sauvegarde d'une table :

Le programme de sauvegarde ci-dessous fonctionne avec ou sans télécommande puisque la sauvegarde de la table se fait en une seule fois.

```
10 '---------- SAUVEGARDE D'UNE TABLE
20 FOR I=1 TO 5
30  A(I)=I
40 NEXT I
50 '
60 PRINT "APPUYEZ SUR <RECORD> PUIS <RETURN>
70 C$=INPUT$(1)
80 '
90 OPEN "CAS:TB" FOR OUTPUT AS #1
100 '
110 FOR I=1 TO 5
120  PRINT #1,A(I)
130 NEXT I
140 CLOSE #1
150 PRINT "ECRITURE TERMINEE"
160 '-------------- LECTURE DE LA TABLE
170 PRINT "POSITIONNEZ ET PLAY"
180 OPEN "CAS:TB" FOR INPUT AS #1
190 '
200 FOR I=1 TO 5
210  INPUT#1,A(I)
220  PRINT A(I)
230 NEXT I
240 CLOSE #1
```

■ Formatage listing :

Le programme ci-dessous permet de lire (comme un fichier) un programme sauvegardé en ASCII (par SAVE) et ainsi de l'éditer formaté à la largeur désirée. La télécommande doit être connectée.

```
10 '------ FORMATTAGE LISTING
15 '
20 INPUT "NOM PROGRAMME ";NP$
30 LG=70                          ' largeur edition
40 OPEN "CAS:"+NP$ FOR INPUT AS #1
50 '
60 IF EOF(1)=-1 THEN END
70 '
80 LINE INPUT#1,LIG$
90 IF LEN(LIG$)<LG THEN LPRINT LIG$:GOTO 60
100 LPRINT LEFT$(LIG$,LG)
110 LPRINT RIGHT$(LIG$,LEN(LIG$)-LG)
120 GOTO 60
```

Ci-dessous, nous utilisons l'éditeur BASIC pour composer un texte.
Nous sauvegardons en ASCII le texte.

```
Ok
10 '            Cher Monsieur
20 '
30 '            BLABLA

save "LETTRE"
Ok
```

Pour lister le texte sans les numéros de ligne, nous utilisons le programme ci-dessous.

```
10 INPUT "Nom texte ";NT$
20 '
30 OPEN NT$ FOR INPUT AS #1
40 '
50 IF EOF(1)=-1 THEN END
60 INPUT #1,NLIG
70 LINE INPUT #1,LIG$
80 LPRINT RIGHT$(LIG$,LEN(LIG$)-1)
90 GOTO 50
Ok
RUN
Nom texte ? LETTRE
```

PROGRAMMES | 11

JOUEZ EN BASIC MSX

- Le squash
- Conduite de voiture
- Bombardement d'immeubles
- Sauts d'obstacles
- Composition de paysage avec animation
- Biorythmes

PROGRAMMES DE GESTION

- Tracé de courbe
- Fichier d'adresses
- Gestion de Fichiers
- Saisie d'écran
- Histogramme
- Histogramme en 3 dimensions
- Bibliothèque

POSSIBILITÉS GRAPHIQUES DU MSX

- Dessin
- Tracé d'un dessin par segments de droite et digitalisation d'un dessin
- Tracé d'un dessin défini en data
- Dessinateur
- Interrogation de géographie

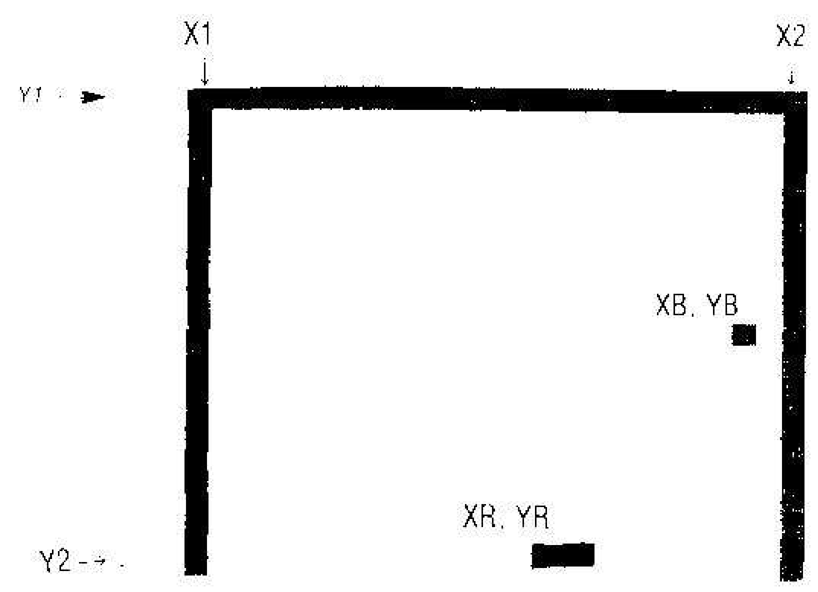
JOUEZ EN MSX

JEU DE SQUASH

Vous devez faire rebondir une balle à l'aide d'une raquette que vous déplacez avec les 2 flèches ← →.
La fonction STICK(0) permet d'obtenir un déplacement plus rapide que la fonction INKEY$; il n'y a pas le délai de répétition.
On remarquera la boucle :

330 FOR V=1 TO 4-NV

Elle règle la vitesse de la balle tout en permettant de tester le clavier.




```
10 SCREEN 0:COLOR 4,15
20 INPUT "Niveau(1,2,3) ";NV
30 '------------------- Dessin terrain
40 X2=19+NV*2
50 X1=1:Y1=1:Y2=19              ' limites terrain
60 KEY OFF:CLS
70 LOCATE 1,21:PRINT "<- -> pour raquette"
80 FOR X=X1 TO X2:LOCATE X,Y1:PRINT CHR$(200):NEXT X
90 FOR Y=Y1 TO Y2
100  LOCATE X1,Y:PRINT CHR$(200)
110  LOCATE X2,Y:PRINT CHR$(200)
120 NEXT Y
130 '
140 RB=0                          ' rebonds
150 DX=1:DY=-1                    ' deplacements
160 XB=5+INT(RND(1)*5):YB=10      ' balle
170 LOCATE XB,YB:PRINT CHR$(200)
180 XR=10:YR=Y2+1                 ' raquette
190 R0$=CHR$(32)+CHR$(200)+CHR$(200)+CHR$(200)+CHR$(32)
200 LOCATE XR-1,YR:PRINT R0$
210 '--------------------- deplacement balle
220 LOCATE XB,YB:PRINT CHR$(32)      ' effacement balle
230 XB=XB+DX:YB=YB+DY                ' nouvelle position
240 LOCATE XB,YB:PRINT CHR$(200)
```

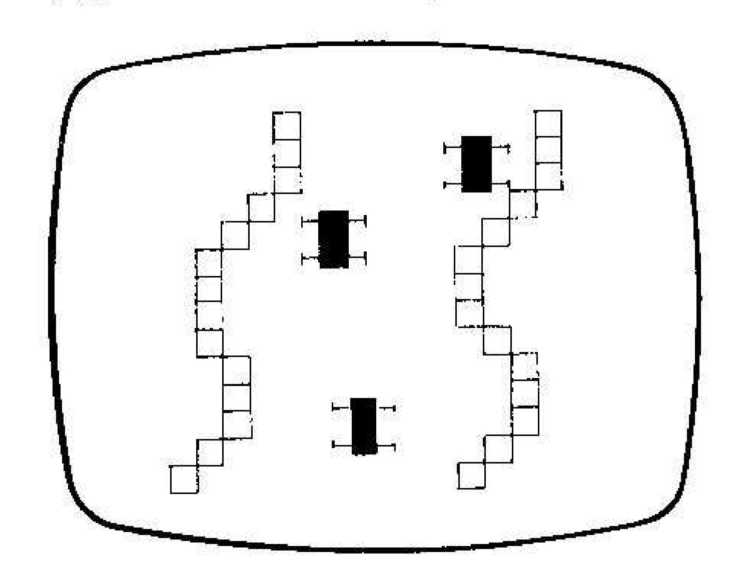
→

```
250 IF XB=>X2-1 THEN DX=-DX          ' rebonds
260 IF YB<Y1+2 THEN DY=-DY
270 IF XB<X1+2 THEN DX=-DX
280 '
290 IF XB>XR-2 AND XB<XR+4 AND YB>YR-2 THENDY=-1 : RB=RB+1
300 '
310 IF YB=>YR THEN LOCATE 25,22:PRINT RB;"Points":GOTO 410
320 '------------------------ deplacement raquette
330 FOR V=1 TO 4-NV                 ' vitesse
340 C=STICK(0):IF C=0 THEN 390
350 '
360 IF C=3 THEN IF XR<X2-3 THEN XR=XR+1
370 IF C=7 THEN IF XR>X1+1 THEN XR=XR-1
380 LOCATE XR-1,YR:PRINT RQ$
390 NEXT V
400 GOTO 220
410 FOR TP=1 TO 1500:NEXT TP:GOTO 40
```

CONDUITE DE VOITURE

En utilisant les flèches ← et →, vous devez éviter de toucher le bord de la route ainsi que les autres véhicules.

La fonction STICK(0) permet d'obtenir un déplacement rapide du véhicule.



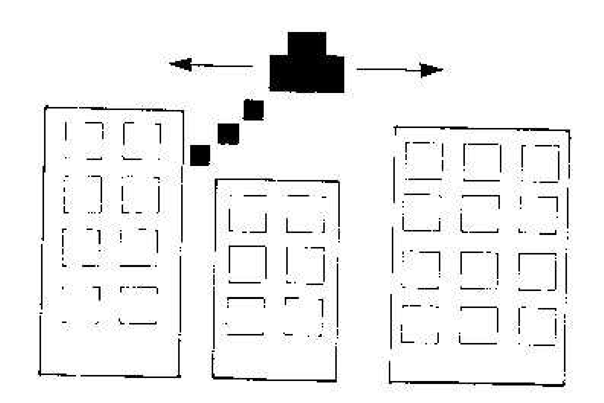
```
10 '-------------- CONDUITE VOITURE
20 '
30 '---- dessin voiture
40 DATA 180,252,180,48,48,180,252,180
50 '
60 SCREEN 0:COLOR 4,15:KEY OFF
70 V$="!"                  ' caractere a modifier
80 AG=BASE(2)              ' adresse generateur de caracteres
90 AC=AG+ASC(V$)*8         ' adresse caractere
100 '
110 FOR M=AC TO AC+7:READ ND:VPOKE M,ND:NEXT M
120 '-------------
130 L$=CHR$(200)+CHR$(200)+"       .       "+CHR$(200)+CHR$(200)
140 AM=BASE(0)                ' adresse memoire ecran
150 '----
```

→

```
160 CLS:LOCATE 1,1:PRINT "FLECHES <- ET ->"
170 INPUT "APPUYER SUR RETURN.";X$
180 XX=10                              ' bord gauche piste
190 XV=XX+5:YV=2                       ' vehicule
200 AX=XV:AY=YV                        ' ancienne position vehicule
210 FOR KM=1 TO 10000
220  X=INT(RND(1)*3)-1                 ' -1,0,+1
230  XX=XX+X
240  IF XX>23 THEN XX=XX-1
250  IF XX<5 THEN XX=XX+1
260  LOCATE XX,23:PRINT L$              ' bas ecran SCROLL
270  IF KM>30 THEN IF RND(1)<.3 THEN LOCATE XX+RND(1)*5+4,21:PRINT V$
280  IF VPEEK(HM+YV*40+XV+2)=ASC(V$) THEN 370  ' test col vehicule
290  IF VPEEK(HM+YV*40+XV+2)=200 THEN 370      ' test col bord piste
300  LOCATE AX,AY-1:PRINT CHR$(32):LOCATE XV,YV:PRINT V$
310  AX=XV:AY=YV
320  C=STICK(0):IF C=0 OR KM<10 THEN 350   ' test clavier
330  IF C=7 THEN XV=XV-1                 ' gauche
340  IF C=3 THEN XV=XV+1                 ' droite
350 NEXT KM
360 '---
370 COLOR 6,1:FOR TP=1 TO 200:NEXT TP:COLOR 4,15
380 PLAY "DEF":LOCATE 1,21:PRINT KM;"POINTS"
390 FOR TP=1 TO 1500:NEXT TP
400 GOTO 160
410 '-----
420 '            ligne 130 4 espaces+"."+4 espaces
```

BOMBARDEMENT D'IMMEUBLES

En appuyant sur "ESPACE", vous bombardez des immeubles.
La soucoupe descend progressivement et ne doit pas s'écraser sur les immeubles.

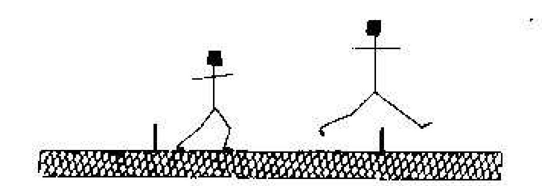


Quatre projectiles peuvent être envoyés simultanément.

```
10 '------ BOMBARDEMENT IMMEUBLES
20 '
30 CLS:INPUT "Niveau (1,2,3 ) ";NV
40 '----- 3 caracteres redefinis(immeuble/soucoupe,projectiles)
50 DATA 252,132,132,132,132,132,132,252
60 DATA 48,48,48,120,252,252,252,252
70 DATA 0,0,0,240,0,0,0,0
80 '
90 SCREEN 0:COLOR 6,15:KEY OFF
100 C$="!"                  '  1er caractere a modifier
110 AG=BASE(2)              ' adresse generateur de caracteres
120 AC=AG+ASC(C$)*8         ' adresse 1er caractere redefini
130 '
140 FOR N=1 TO 3            ' 3 caracteres
150  A=AC+(N-1)*8
160  FOR M=A TO A+7:READ ND:VPOKE M,ND:NEXT M
170 NEXT N
180 '--
190 C=ASC(C$)
200 '------------- DESSIN IMMEUBLES
210 XB=2:YB=21:CLS
220 FOR N=1 TO 5
230  H=RND(1)*8+4+NV:L=RND(1)*3+3
240  FOR Y=YB-H TO YB
250   FOR X=XB TO XB+L
260    LOCATE X,Y:PRINT CHR$(C);
270   NEXT X
280  NEXT Y
290 XB=XB+L+2
300 NEXT N
310 LOCATE 1,1:PRINT "APPUYER SUR <ESPACE>"
320 '================================ BOMBARDEMENT
330 AM=BASE(0)                    ' adresse ecran
340 YS=2:X1=1:X2=37:S=1           ' soucoupe/bornes/sens
350 SOUND 8,12:SOUND 1,0:SOUND 7,254
360 FOR XS=X1 TO X2 STEP S
370  IF YS>16 THEN SOUND 7,255:GOTO 210
380  IF VPEEK(AM+2+XS+YS*40)=ASC(C$) THEN 580
390  LOCATE XS-S,YS:PRINT CHR$(32);
400  LOCATE XS,YS:PRINT CHR$(C+1);
410  SOUND 0,XS*2
420  IF INKEY$="" THEN 470     ' depart projectile
430  FOR I=1 TO 4
440   IF YP(I)=0 THEN YP(I)=YS+1:XP(I)=XS:GOTO 470
450  NEXT I
460 '
470  FOR I=1 TO 4                  ' avance projectiles
480   IF YP(I)=0 THEN 520
490   LOCATE XP(I),YP(I):PRINT CHR$(32);
500   YP(I)=YP(I)+1:LOCATE XP(I),YP(I):PRINT CHR$(C+2);
510   IF YP(I)>20 THEN YP(I)=0:GOTO 520
520  NEXT I
530 NEXT XS
540 LOCATE XS-S,YS:PRINT CHR$(32);:SWAP X1,X2:S=-S  ' inversion
550 YS=YS+1
560 GOTO 360
570 '-- collision
580 PLAY "CDE":FOR TP=1 TO 1000:NEXT TP:GOTO 210
```

SAUTS D'OBSTACLES

En appuyant sur "ESPACE", vous devez sauter des obstacles aléatoires.



Deux sprites représentent la position normale et la position de saut.

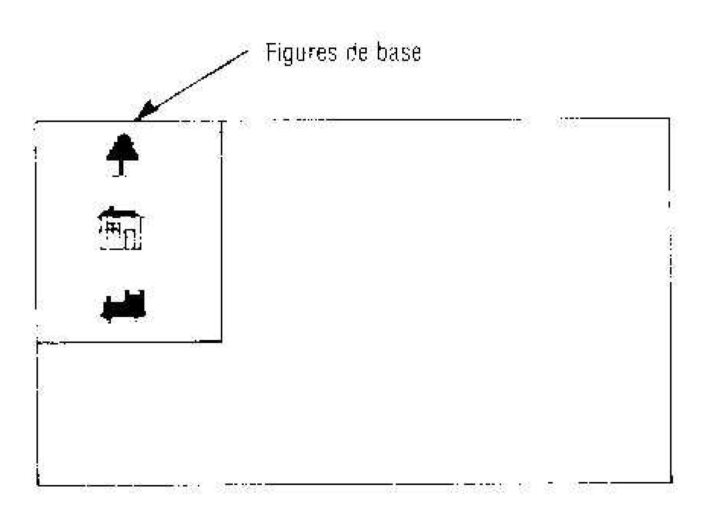
```
10 '============= ====== SAUT D'OBSTACLE
20 CLEAR 1000
30 CLS:INPUT "Niveau (1,2,3,4)";NV
40 CE=1:CF=15                 ' ecriture/fond
50 COLOR CE,CF:SCREEN 2,0
60 FOR S=1 TO 2               ' 2 sprites 16x16
70   FOR I=1 TO 4:A$(I)="":NEXT I
80   FOR N=1 TO 4
90    FOR I=1 TO 8
100     READ X:A$(N)=A$(N)+CHR$(X)
110    NEXT I
120   NEXT N
130   SPRITE$(S)=A$(1)+A$(2)+A$(3)+A$(4)
140 NEXT S
150 '------------------------- Position normale
160 DATA 3,3,3,1,7,13,1,1
170 DATA 3,2,6,4,12,8,12,0
180 DATA 128,128,128,0,0,192,96,0
190 DATA 128,192,96,32,32,32,48,0
200 '------------------------- Position saut
210 DATA 3,3,3,1,15,1,1,1
220 DATA 3,2,6,28,16,0,0,0
230 DATA 128,128,128,0,0,128,0,0
240 DATA 128,192,96,48,24,0,0,0
250 '================================
260 OPEN "GRP:" FOR OUTPUT AS #1
270 DRAW "BM10,160":PRINT #1,"APPUYER SUR <ESPACE>"
280 Y=120          ' obstacle
290 S=5            ' avance
300 LINE (1,Y)-(200,Y),1
310 '----- 2 OBSTACLES AU HASARD
320 XB(1)=INT(RND(1)*12)*S+40:XB(2)=XB(1)+INT(RND(1)*5)*S+40
330 FOR I=1 TO 2:LINE (XB(I),Y)-(XB(I),Y-5),1:NEXT I    ' obstacles
340 '------------------- deplacement
350 FOR X=10 TO 200 STEP S
360  PUT SPRITE 1,(X,Y-16),6,1
370  FOR I=1 TO 2
380   IF X>XB(I)-2*S AND X<XB(I) THEN 420
390  NEXT I
400  GOTO 460
410 '---
420  RT=RT+1:DRAW "BM10,170":COLOR CF:PRINT #1,STRING$(12,CHR$(200))
430  COLOR CE:DRAW "BM10,170":PRINT #1,RT;"RATE(S)"
440  BEEP
450  X=X+S
```

→

```
460  IF INKEY$<>"" THEN GOSUB 530
470  FOR TP=1 TO 20-NV*4:NEXT TP
480 NEXT X
490 FOR I=1 TO 2:LINE(XB(I),Y)-(XB(I)+5),CF:NEXT I
500 PS=PS+1:IF PS=10 THEN C$=INPUT$(1):END
510 GOTO 280
520 '------- SAUT OBSTACLE
530 IF TIME<8 THEN TIME=0:RETURN
540 PUT SPRITE 1,(X,209),6,1
550 PUT SPRITE 2,(X,Y-25),6,2
560 FOR TP=1 TO 300:NEXT TP
570 PUT SPRITE 2,(X,209),6,2
580 X=X+3*S
590 TIME=0
600 RETURN
```

COMPOSITION DE PAYSAGE AVEC ANIMATION

A l'aide de 3 figures de base (arbre, maison, locomotive), vous composez un paysage. Le choix des figures se fait à l'aide d'un curseur que vous déplacez avec les quatre flèches.
En frappant "P", vous "prenez" une des 3 figures de base (BEEP signale que la figure a été choisie).
"D" permet de "déposer" la figure choisie.
La couleur se choisit en frappant 1,2,3,..9.
Pour animer une figure, placer le curseur devant elle puis frapper "+" ou "−" (une ou plusieurs fois).
Une figure animée peut également être stoppée.




```
10 '-------------- COMPOSITION PAYSAGE  AVEC ANIMATION
20 CE=1:CF=15:COLOR CE,CF
30 SCREEN 2,2:CL=CE
40 '
50 XC=8*10:YC=8*10        ' curseur
60 DIM X(32),Y(32),V(32)
70 NS=3                   ' nombre de figures de base
80 NN=NS                  ' nombre de figures
90 OPEN "GRP:" FOR OUTPUT AS #1
100  PRESET(15,145):PRINT #1,"CLAVIER MAJUSCULE"
110 PSET(15,155):PRINT #1,"FLECHES PUIS:"
120 PSET(15,165):PRINT #1,"P:PRENDRE/  D:DEPOSER "
```

```
130 PSET(15,175):PRINT #1,"COUL: 1,2,3,.. + - VITESSE"
140 '----------------------- ARBRE
150 DATA 1,3,7,7,15,15,15,31
160 DATA 63,63,63,3,3,3,3,15
170 DATA 0,128,192,192,224,224,240,240
180 DATA 248,248,248,0,0,0,0,192
190 '------------------------- MAISON
200 DATA 12,12,63,127,255,64,95,85
210 DATA 95,85,85,85,95,64,64,127
220 DATA 0,0,248,252,255,2,2,2
230 DATA 122,74,74,106,74,74,74,254
240 '--------------------- LOCOMOTIVE
250 DATA 0,0,6,0,16,0,48,48
260 DATA 127,127,127,127,255,255,48,48
270 DATA 0,0,0,0,254,198,254,254
280 DATA 254,254,254,254,255,255,24,24
290 '----
300 FOR S=1 TO NS
310   FOR I=1 TO 4:S$(I)="":NEXT I
320   FOR N=1 TO 4
330    FOR I=1 TO 8
340     READ X:S$(N)=S$(N)+CHR$(X)
350    NEXT I
360   NEXT N
370   SPRITE$(S)=S$(1)+S$(2)+S$(3)+S$(4)
380 NEXT S
390 '-------Affichage sprites de base
400 FOR I=1 TO NS
410  X(I)=16:Y(I)=I*24
420  PUT SPRITE I,(X(I),Y(I)),4,I
430 NEXT I
440 '------------------- AIGUILLAGE
450 GOSUB 620
460 IF C$="P" THEN GOSUB 810
470 IF C$="D" THEN GOSUB 890
480 GOTO 450
```

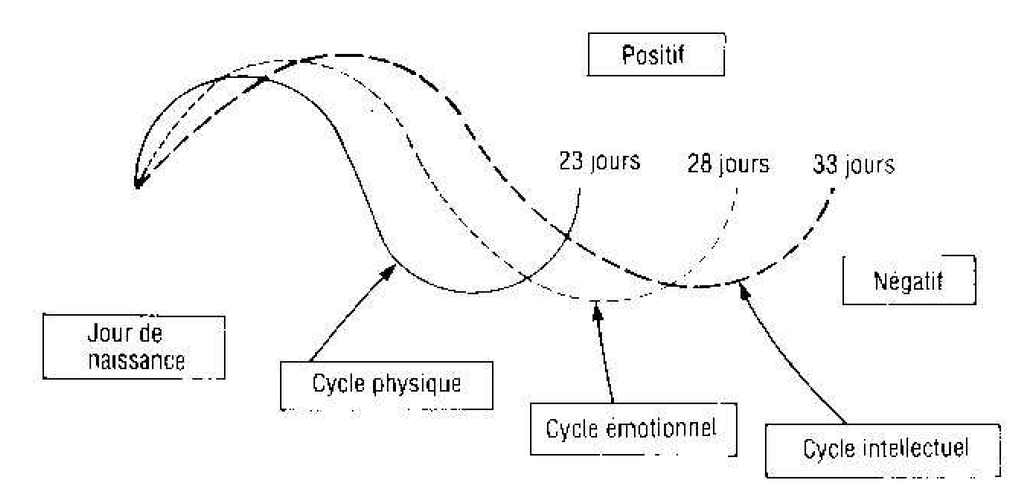
```
610 '----------------- GESTION CURSEUR
620 C$=INKEY$:IF C$<>"" THEN 680
630 LINE(XC,YC)-(XC+4,YC),CE
640 LINE(XC,YC)-(XC+4,YC),CF
650 GOSUB 970
660 GOTO 620
670 '-
680 C=ASC(C$)
690 IF C=28 THEN IF XC<240 THEN XC=XC+8
700 IF C=29 THEN IF XC>16 THEN XC=XC-8
710 IF C=30 THEN IF YC>10 THEN YC=YC-8
720 IF C=31 THEN IF Y<180 THEN YC=YC+8
730 IF C$="P" THEN RETURN
740 IF C$="D" THEN RETURN
750 IF VAL(C$)>0 THEN CL=VAL(C$)
760 IF C$="+" THEN V=1:GOSUB 1060
770 IF C$="-" THEN V=2:GOSUB 1060
780 GOTO 620
```

→

```
790 '--------------- ON PREND
800 ' BEEP signale que la figure est choisie.
810 SS=0
820 FOR I=1 TO NS
830  IF XC=>X(I) AND XC<=X(I)+8  AND YC=>Y(I) AND YC<=Y(I)+8
   THEN 860
840 NEXT I
850 RETURN
860 SS=I:BEEP
870 RETURN
880 '------------------- ON POSE
890 IF SS=0 THEN RETURN
900 IF NN>30 THEN RETURN
910 IF XC<35 THEN RETURN
920 NN=NN+1:SPRITE$(NN)=SPRITE$(SS)
930 PUT SPRITE NN,(XC,YC),CL,NN
940 Y(NN)=YC:X(NN)=XC
950 RETURN
960 '------------------- AVANCE
970 FOR I=NS+1 TO NN
980  IF V(I)=0 THEN 1030
990  X(I)=X(I)-V(I)
1000  IF X(I)<35 THEN X(I)=220
1010  IF X(I)>220 THEN X(I)=35
1020  PUT SPRITE I,(X(I),Y(I)),,I
1030 NEXT I
1040 RETURN
1050 '------------------- REGLAGE VITESSE
1060 FOR I=NS+1 TO NN
1070  IF YC=>Y(I) AND YC<=Y(I)+16 THEN 1110
1080 NEXT I
1090 RETURN
1100 '
1110 IF V=1 THEN IF XC>30 THEN V(I)=V(I)+1:V=0
1120 IF V=2 THEN IF XC>30 THEN V(I)=V(I)-1:V=0
1130 RETURN
```

BIORYTHMES

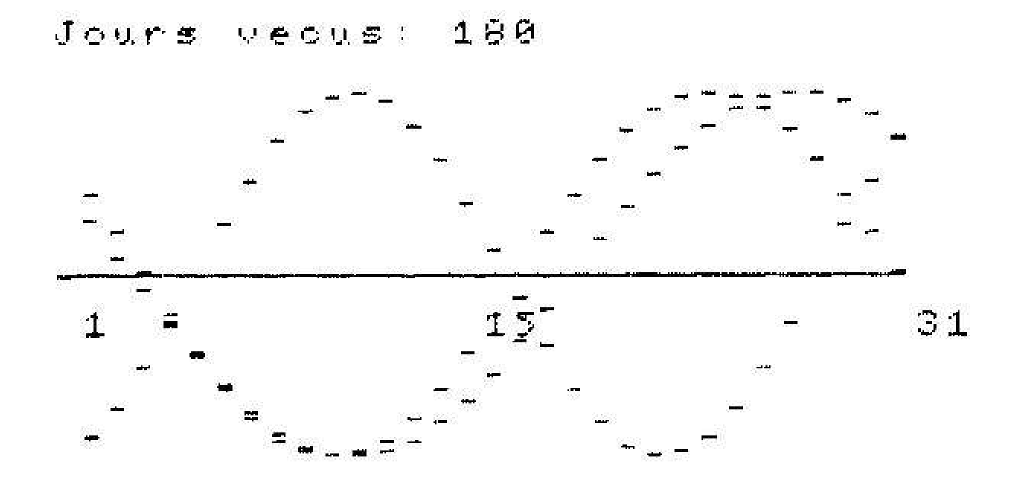
Il existerait chez l'homme des périodes "positives" et des périodes "négatives" réglées par des "horloges internes" indépendantes de l'environnement extérieur.



Il y aurait trois types de cycles :

- Cycle physique de 23 jours ;
- Cycle émotionnel de 28 jours ;
- Cycle intellectuel de 33 jours.

On pourrait ainsi connaître à l'avance les jours favorables.

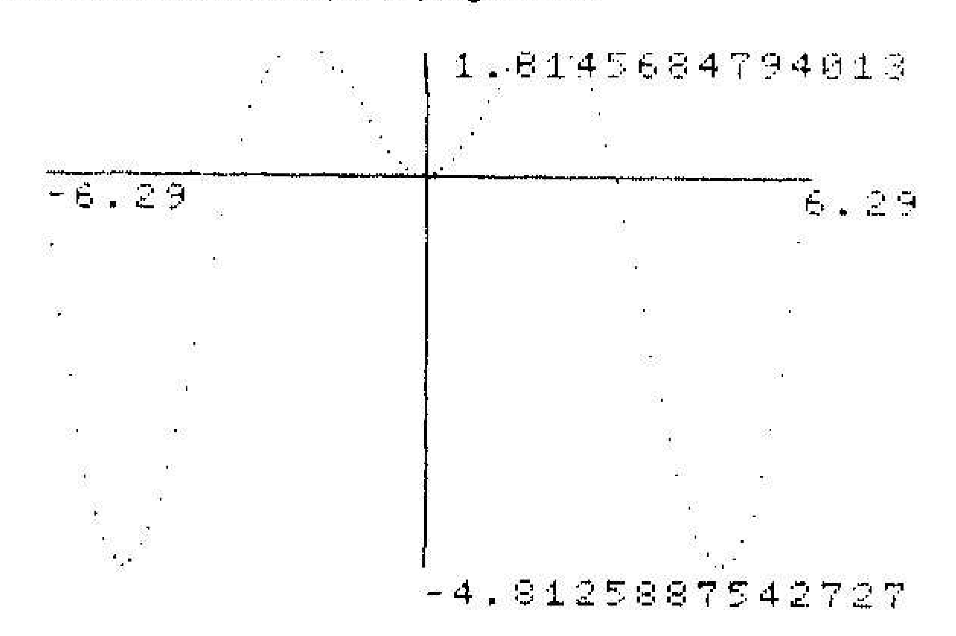


```
10 '--------------- BIORYTHME
20 PI=3.14159
30 DIM A(12),JM(12)
40 '
50 DATA 31,28.25,31,30,31,30,31,31,30,31,30,31
60 FOR I=1 TO 12:A(I)=A(I-1):READ X:A(I)=A(I)+X:
   JM(I)=X:NEXT I
70 CP=23:CE=28:CI=33
80 '
90 SCREEN 0
100 PRINT "NAISSANCE:"
110 INPUT "JOUR,MOIS,AN (EX:5,4,1984) ";JN,MN,AN
120 IF AN<1900 OR AN>1999 THEN 110
130 J=JN:M=MN:A=AN:GOSUB 210:XV=JV
140 '
150 PRINT "BIORYTHME:"
160 INPUT "MOIS,AN (EX:10,1984) ";MB,AB
170 IF AB<1900 OR AB>1999 THEN 160
180 J=1:M=MB:A=AB:GOSUB 210:NJOUR=JV-XV
190 GOTO 240
200 '-------------------- CALENDRIER
210 N=365.25*(A-1901)+A(M)+J
220 JV=INT(N)
230 RETURN
235 '------------------- AFFICHAGE COURBES
240 COLOR 1,15:SCREEN 2
250 SCREEN 2
260 OPEN "GRP:" FOR OUTPUT AS #1
270 PRESET(10,160):PRINT #1,"BLEU:PHYSIQUE"
280 PRESET(10,170):PRINT #1,"NOIR:EMOTIONNEL"
290 PRESET(10,180):PRINT #1,"ROUGE:INTELLECTUEL"
300 PRESET(10,90):PRINT #1,1;SPC(10);15;SPC(10);31
310 LINE (10,80)-(230,80),1
320 K=0
330 FOR D=1 TO JM(MB)
340  P1=50*SIN((K+NJOUR)*2*PI/CP)
350  P2=50*SIN((K+NJOUR)*2*PI/CE)
360  P3=50*SIN((K+NJOUR)*2*PI/CI)
370  X=D*7+10
380  LINE (X,80-P1)-(X+3,80-P1),4
390  LINE (X,80-P2)-(X+3,80-P2),1
400  LINE (X,80-P3)-(X+3,80-P3),6
410  K=K+1
420 NEXT D
430 PRESET(10,10):PRINT #1,"Jours vecus:";NJOUR
440 C$=INPUT$(1)
```

PROGRAMMES DE GESTION

TRACÉ DE COURBE

Ce programme trace la courbe d'une fonction écrite en 400.
Les échelles sont calculées par le programme.



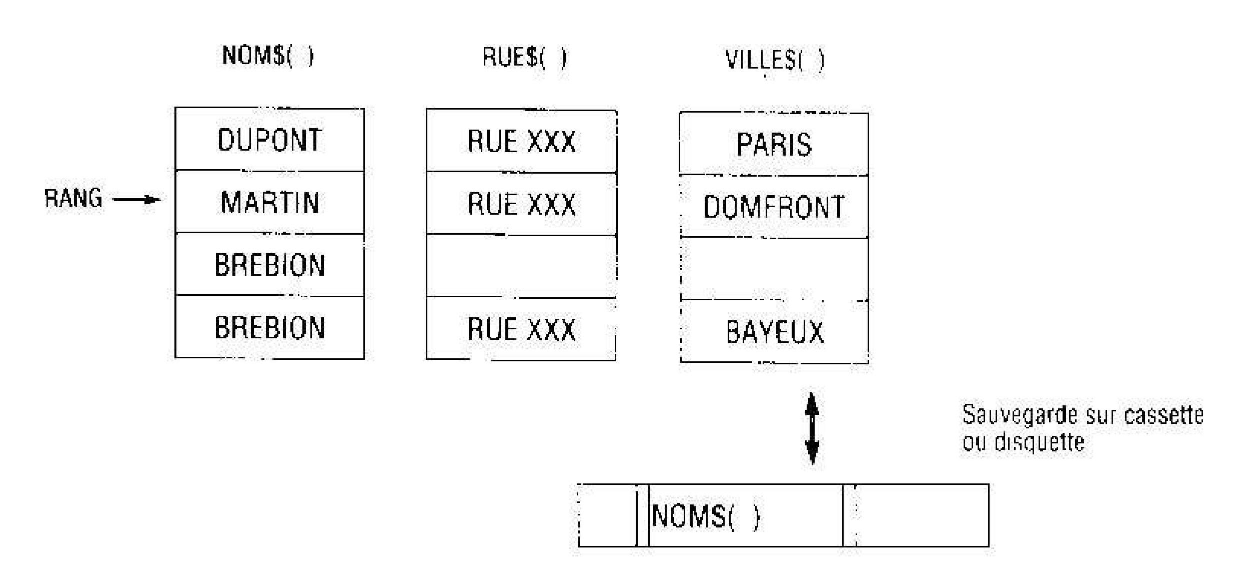

```
10 '--------------- TRACE DE COURBE
20 '
40 HECR=150:LECR=200     ' hauteur/longueur ecran
50 INPUT "Borne x1 ";B1
60 INPUT "Borne x2 ";B2
70 INPUT "Pas ";PAS
80 SCREEN 2
100 '----------- recherche mini/maxi y
110 X=B1:GOSUB 400:Y1=Y:Y2=Y
120 FOR X=B1 TO B2 STEP PAS
130  GOSUB 400
140  IF Y<Y1 THEN Y1=Y
150  IF Y>Y2 THEN Y2=Y
160 NEXT X
170 EX=LECR/(B2-B1)
180 EY=(HECR-2)/(Y2-Y1)
190 '------------------- AXE Y
200 IF B2=>0 AND B1<=0 THEN X=-EX*B1:LINE (X,1)-(X,HECR),1
210 '-------------------- AXE X
220 IF Y2=>0 AND Y1<=0 THEN Y=HECR+Y1*EY:LINE (1,Y)-(LECR,Y),1
230 '-------------------- COURBE
240 FOR X=B1 TO B2 STEP PAS
250 GOSUB 400
260  SX=(X-B1)*EX
270  SY=HECR-(Y-Y1)*EY
280  PSET(SX,SY),1
290 NEXT X
300 '----------- AFFICHAGE EXTREMES
```

→

```
305 OPEN "GRP:" FOR OUTPUT AS #1
310 Y=HECR-ABS(Y1)*EY+3:X=1:B=B1:GOSUB 370
320 Y=HER-ABS(Y1)*EY+3:X=LECR-9:B=B2:GOSUB 370
330 Y=HECR+4:X=ABS(B1)*EX:B=Y1:GOSUB 370
340 Y=5:X=ABS(B1)*EX:B=Y2:GOSUB 370
350 C$=INPUT$(1)
355 END
360 '--
370 IF X>LECR+10 THEN X=LECR/2
371 IF Y>HECR+10 THEN Y=HECR/2
375 DRAW "BM=X;,=Y;":PRINT #1,B
380 RETURN
390 '------------------ courbe a tracer
400 Y=SIN(X)*X
410 RETURN
```

FICHIER D'ADRESSES

Le programme ci-dessous permet d'introduire et de modifier des données indépendantes du programme. Elles sont temporairement stockées dans des tables qui sont ensuite sauvegardées sur cassette.

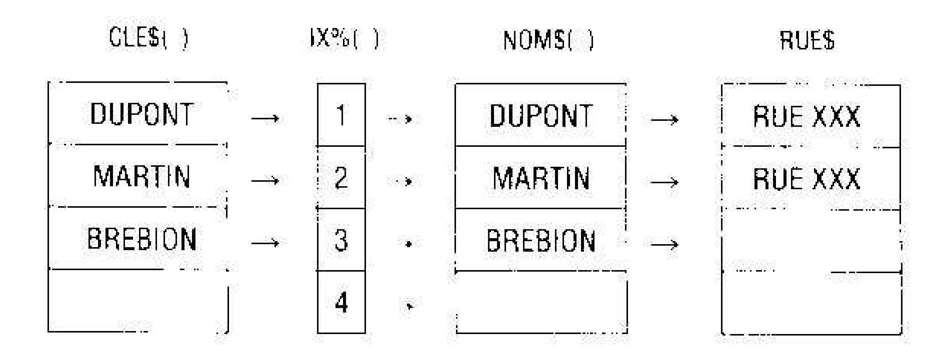


La variable "RANG" donne l'adresse de rangement dans les tables.
Le mode "C" permet à la fois de créer et de modifier des fiches.
En mode modification, la valeur de chaque zone est affichée puis le programme attend une nouvelle valeur.
Si vous ne voulez pas modifier une zone, appuyez sur "RETURN" sans entrer de valeur.
L'utilisation de l'instruction LINE INPUT (au lieu de INPUT) permet d'introduire le caractère virgule dans les zones.
L'instruction 310 peut être remplacée par :

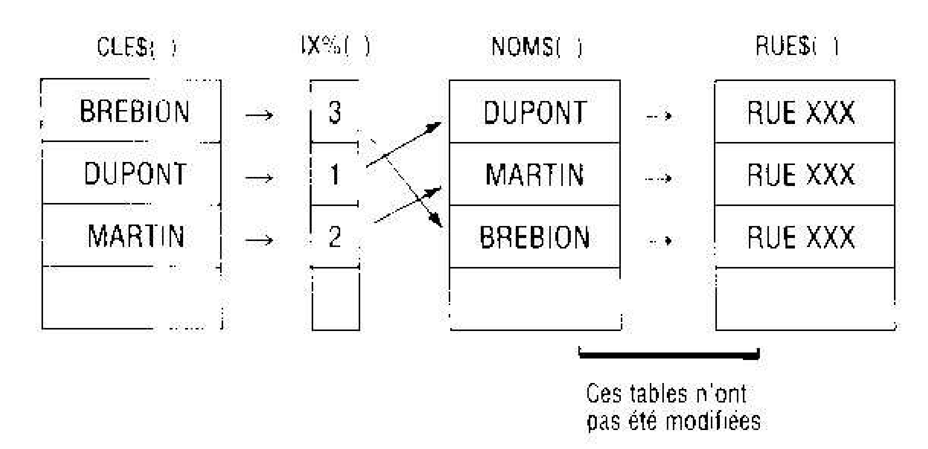
```
310 IF NOM$=LEFT$-NOM$(RANG),LEN(NOM$) THE 390
```

On peut ainsi entrer seulement les premières lettres du nom.

Pour obtenir une **liste triée par noms,** nous remplissons une "table des Clés" CLE$() avec les clés à trier.
Dans une "table d'index" IX%(), nous rangeons les numéros de ligne (1, 2, 3,...).



Nous trions les tables CLE$() et IX%()



Après le tri, il suffit de lire la table IX%() pour éditer les adresses dans l'ordre des noms.

TRI-SÉLECTION

Pour obtenir la liste triée des personnes d'une seule ville, il suffit de sélectionner les noms ainsi :

```
INPUT « Quelle ville », V$
IF V$<> VILLE$(F) THEN ...
```

TRI-MULTICRITÈRES

La liste alphabétique des clients dans l'ordre des villes s'obtient en faisant :

```
CLE$(F) = VILLE$(F) + NOM$(F)
```

au lieu de :

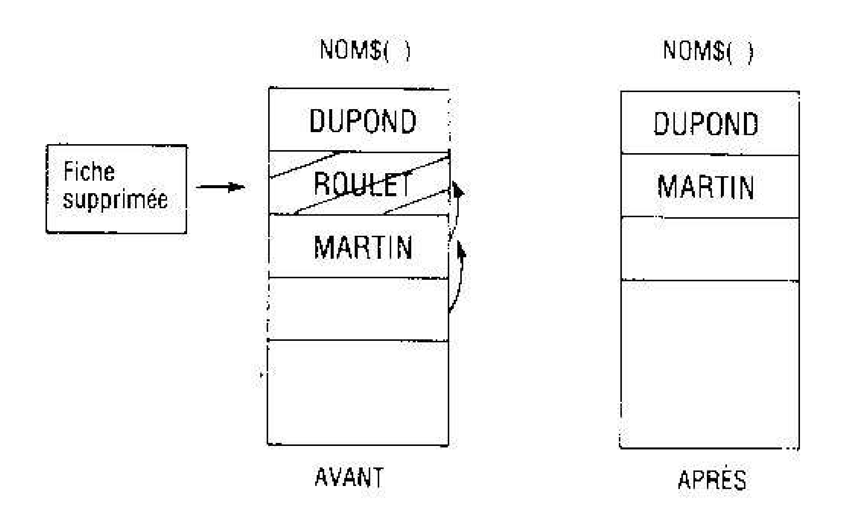
```
CLE$(F) = NOM$(F)
```

CLE$( )

BOULOGNE MARTIN
BOULOGNE VINCENT
PARIS DUPONT
PARIS DURAND

SUPPRESSION

Pour supprimer une fiche, nous décalons toutes les fiches en aval de la fiche supprimée.
L'instruction SWAP nous permet de décaler les éléments sans provoquer de réorganisation de l'espace chaîne.




```
10 '--------------- FICHIER D'ADRESSES
20 '
30 CLEAR 5000
40 DIM NOM$(100),RUE$(100),VILLE$(100),CPST$(100)
50 DIM CLE$(100),IX%(100)
60 NFICH=0              ' nombre de fiches
70 CLS:INPUT "Nouveau fichier (O/N) ";R$:IF R$="O" THEN 100
80 GOSUB 770
90 '
100 CLS:PRINT "MODES:":PRINT
110 PRINT TAB(3);"C   :CREATION/MODIFICATION"
120 PRINT TAB(3);"LF  :LISTE DU FICHIER"
130 PRINT TAB(3);"FIN :FIN DE SESSION(SAUVEGARDE)"
140 PRINT TAB(3);"LFN :LISTE DU FICHIER PAR NOM"
150 PRINT TAB(3);"LFV :LISTE DU FICHIER PAR VILLE"
160 PRINT:M$="":INPUT "MODE ";M$
170 IF M$="C" THEN GOSUB 260
180 IF M$="FIN" THEN GOSUB 640
190 IF M$="LF" THEN GOSUB 920
200 IF M$="LFN" THEN GOSUB 1030
210 IF M$="LFV" THEN GOSUB 1400
220 IF M$="S" THEN GOSUB 1540
230 GOTO 100
240 '======================= CREATION ET MODIFICATION =======
250 '
```

```
260 PRINT
270 LINE INPUT "NOM(OU <RETURN>)? ";NOM$:IF LEN(NOM$)=0 THEN
    RETURN
280 '
290 IF NFICH=0 THEN 340
300 FOR RANG=1 TO NFICH
310  IF NOM$=NOM$(RANG) THEN 390      ' nom existe t-il?
320 NEXT RANG
330 '---------- nouveau nom
340 PRINT:R$="" INPUT "Nouveau nom OK (O/N) ";R$:IF R$<>"O"
    THEN 260
350 NFICH=NFICH+1
360 RANG=NFICH
370 NOM$(RANG)=NOM$
380 '---------- entree/modif zones
390 PRINT
400 PRINT RUE$(RANG);TAB(15);          ' ancienne valeur
410 LINE INPUT 'Rue? ";RUE$:IF RUE$<>"" THEN RUE$(RANG)=RUE$
420 PRINT VILLE$(RANG);TAB(15);
430 LINE INPUT "Ville? ";VILLE$
440 IF VILLLE$<>"" THEN VILLE$(RANG)=VILLE$
450 PRINT CPST$(RANG);TAB(15);
460 LINE INPUT "Code Postal? ";CPST$
470 IF CPST$<>"" THEN CPST$(RANG)=CPST$
480 GOTO 260
490 '------
500 ' l'instruction 310  Peut etre changee Par:
510 ' 310 IF NOM$=LEFT$(NOM$,LEN(NOM$) THEN 390
520 ' ceci Permet d'entrer seulement les Premieres lettres du
     NOM
630 '======================== SAUVEGARDE
640 PRINT:PRINT "APPUYER SUR <RECORD> PUIS <RETURN>"
650 X$=INPUT$(1)
660 OPEN "CAS:ADR" FOR OUTPUT AS #1
670 '
680 FOR I=1 TO NFICH
690  PRINT #1,NOM$(I)
700  PRINT #1,RUE$(I)
710  PRINT #1,VILLE$(I)
720  PRINT #1,CPST$(I)
730 NEXT I
740 CLOSE#1
750 RETURN
760 '======================= LECTURE FICHIER
770 PRINT:PRINT "APPUYEZ SUR <PLAY>":PRINT
780 OPEN "CAS:ADR" FOR INPUT AS #1
790 '
800 FOR I=1 TO 100
810  IF EOF(1)=-1 THEN NFICH=I-1:CLOSE #1:GOTO 880
820  LINE INPUT#1,NOM$(I)
830  LINE INPUT#1,RUE$(I)
840  LINE INPUT#1,VILLE$(I)
850  LINE INPUT#1,CPST$(I)
860 NEXT I
870 STOP
880 PRINT:PRINT NFICH;"NOMS"
890 FOR TP=1 TO 1000:NEXT TP
900 RETURN
910 '==================== LISTE DU FICHIER
920 CLS
```

```
930 PRINT "LISTE DU FICHIER":PRINT
940 '
950 FOR F=1 TO NFICH
960  PRINT NOM$(F);TAB(13);
970  PRINT VILLE$(F);
980  PRINT
990 NEXT F
1000 PRINT:INPUT "APPUYEZ SUR <RETURN>";X$
1010 RETURN
1020 '======================== LISTE TRIEE PAR NOM
1030 FOR F=1 TO NFICH
1040  CLE$(F)=NOM$(F):IX%(F)=F
1050 NEXT F
1060 '
1070 NCL=NFICH
1080 GOSUB 1280      ' appel tri
1090 '
1100 CLS:PRINT "LISTE TRIEE PAR NOM":PRINT
1110 FOR F=1 TO NCL
1120  X=IX%(F)
1130  PRINT NOM$(X);TAB(15);
1140  PRINT VILLE$(X)
1150 NEXT F
1160 PRINT:INPUT "APPUYER SUR <RETURN> ";X$
1170 RETURN
1270 '------ TRI SHELL
1280 ECART=NCL
1290 ECART=INT(ECART/2):IF ECART<1 THEN RETURN
1300 IV=0
1310 FOR I=1 TO NF-ECART
1320  J=I+ECART
1330  IF CLE$(J)>=CLE$(I) THEN 1360
1340  SWAP CLE$(I),CLE$(J):IV=1
1350  SWAP IX%(I),IX%(J)
1360 NEXT I
1370 IF IV=1 THEN 1300
1380 GOTO 1290
1390 '================== LISTE TRIEE PAR VILLE
1400 FOR F=1 TO NFICH
1410  CLE$(F)=VILLE$(F):IX%(F)=F
1420 NEXT F
1430 NCL=NFICH
1440 GOSUB 1280       ' appel tri
1450 CLS:PRINT "LISTE TRIEE PAR VILLES":PRINT
1460 FOR F=1 TO NCL
1470  X=IX%(F)
1480  PRINT VILLE$(X);TAB(16);
1490  PRINT NOM$(X)
1500 NEXT F
1510 PRINT :INPUT "APPUYER SUR <RETURN>";X$
1520 RETURN
1530 '=========================== SUPPRESSION
1540 PRINT:NOM$="":INPUT "NOM ";NOM$:IF NOM$="" THEN RETURN
1550 '
1560 FOR RANG=1 TO NFICH
1570 IF NOM$(RANG)=NOM$ THEN 1620
1580 NEXT RANG
1590 PRINT:PRINT "N'EXISTE PAS":PRINT:GOTO 1540
1600 '
1610 PRINT
```

```
1620 R$="":INPUT "SUPPRESSION OK (O/N) ";R$:IF R$<>"O" THEN
     1540
1630 FOR J=RANG TO NFICH-1
1640  SWAP NOM$(J),NOM$(J+1)
1650  SWAP RUE$(J),RUE$(J+1)
1660  SWAP VILLE$(J),VILLE$(J+1)
1670  SWAP CPST$(J),CPST$(J+1)
1680 NEXT J
1690 NOM$(NFICH)="":RUE$(NFICH)="":VILLE$(NFICH)="":CPST$
     (NFICH)=""
1700 NFICH=NFICH-1
1710 GOTO 1540
```

```
          C    :CREATION/MODIFICATION
          LF   :LISTE DU FICHIER
          FIN  :FIN DE SESSION(SAUVEGARDE)
          LFN  :LISTE DU FICHIER PAR NOM
          LFV  :LISTE DU FICHIER PAR VILLE

      MODE ? C

      NOM(OU <RETURN>)? DUPONT

      Nouveau nom OK (O/N) ? O

                        Rue? 11,RUE NOBEL
                        Ville? MONTIGNY
                        Code Postal? 78180

      NOM(OU <RETURN>)?
      Break in 270
```

GESTION DE FICHIER

Nous proposons un programme de gestion de fichier adaptable.
Chaque ligne d'une table FICH$(,) contient un "enregistrement".

		CODE	LIBELLÉ	PRIX	STOCK	TABLE FICH$(,)
	1	R5	XXXXX	50000	20	
RANG →	2	R18	XXXX	60000	12	
	3	R4	XXXXX	400000	30	
	4	R35		700000	4	
	5					

La table est sauvegardée dans un fichier séquentiel sur cassette ou disquette.

```
10 '-------------------- FICHIER
20 '
30 '
40 CLEAR 5000
50 NRUB=4                ' nombre de rubriques(a adapter)
60 MFICH=100             ' nombre maxi de fiches
70 DIM FICH$(MFICH,NRUB)
80 '------------ noms des rubriques(a adapter)
90 RUB$(1)="CODE"
100 RUB$(2)="LIBELLE"
110 RUB$(3)="PRIX"
120 RUB$(4)="STOCK"
130 '---------------------
140 '
150 NFICH=0              ' nombre de fiches
160 CLS:INPUT "Nouveau fichier (O/N) ";R$:IF R$="O" THEN 190
170 GOSUB 710
180 '
190 CLS:PRINT "MODES:":PRINT
200 PRINT TAB(3);"C    :CREATION/MODIFICATION"
210 PRINT TAB(3);"LF   :LISTE DU FICHIER"
220 PRINT TAB(3);"FIN  :FIN DE SESSION(SAUVEGARDE)"
230 PRINT:M$="":INPUT "MODE ";M$
240 IF M$="C" THEN GOSUB 300
250 IF M$="FIN" THEN GOSUB 570
260 IF M$="LF" THEN GOSUB 850
270 GOTO 190
280 '======================== CREATION ET MODIFICATION ========
290 '
300 PRINT
310 PRINT RUB$(1);
320 LINE INPUT "?";CLE$:IF LEN(CLE$)=0 THEN RETURN
330 '
340 IF NFICH=0 THEN 390
350 FOR RANG=1 TO NFICH
360  IF CLE$=FICH$(RANG,1) THEN 440 ' nom existe t-il?
370 NEXT RANG
380 '---------- nouvelle cle
390 PRINT:R$="":INPUT "Nouvelle cle OK (O/N) ";R$:IF R$<>"O"
    THEN 300
400 NFICH=NFICH+1
410 RANG=NFICH
420 FICH$(RANG,1)=CLE$
430 '---------- entree/modif zones
440 PRINT
450 FOR R=1 TO NRUB
460  PRINT FICH$(RANG,R);TAB(15);     ' ancienne valeur
470  PRINT RUB$(R);TAB(22);           ' nom de zone
480  LINE INPUT "?";X$
490  IF X$<>"" THEN FICH$(RANG,R)=X$
500 NEXT R
510 GOTO 300
520 '------
```

```
560 '======================== SAUVEGARDE
570 PRINT:PRINT "APPUYER SUR <RECORD> PUIS <RETURN>"
580 X$=INPUT$(1)
590 OPEN "CAS:FICH" FOR OUTPUT AS #1
600 '
610 PRINT #1,NFICH
620 '
630 FOR I=1 TO NFICH
640  FOR J=1 TO NRUB
650   PRINT #1,FICH$(I,J)
660  NEXT J
670 NEXT I
680 CLOSE#1
690 RETURN
700 '========================= LECTURE FICHIER
710 PRINT:PRINT "APPUYEZ SUR <PLAY>":PRINT
720 OPEN "CAS:FICH" FOR INPUT AS #1
730 '
740 INPUT #1,NFICH
750 FOR F=1 TO NFICH
760  FOR R=1 TO NRUB
770   LINE INPUT#1,FICH$(F,R)
780  NEXT R
790 NEXT F
800 CLOSE #1
810 PRINT:PRINT NFICH;"FICHES"
820 FOR TP=1 TO 1000:NEXT TP
830 RETURN
840 '==================== LISTE DU FICHIER
850 CLS
860 PRINT "LISTE DU FICHIER":PRINT
870 '
880 FOR F=1 TO NFICH
890  PRINT FICH$(F,1);TAB(10);    ' zone 1
900  PRINT FICH$(F,2)             ' zone 2
910 NEXT F
920 PRINT:INPUT "APPUYEZ SUR <RETURN>";X$
930 RETURN
```

SAISIE D'ÉCRAN

Une saisie d'informations avec l'instruction "INPUT" ne permet pas de se positionner sur une zone qui aurait été mal documentée.
Le programme suivant le permet. Il utilise la fonction INPUT$(1).
En outre, les caractères frappés au clavier peuvent être contrôlés dès leur introduction, sans attendre la frappe de "RETURN".

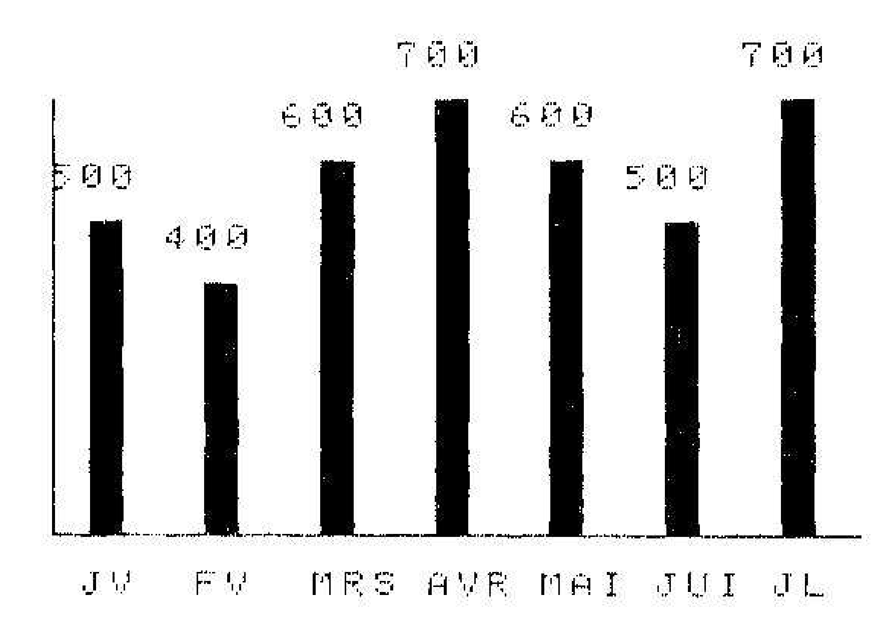
```
NOM:      DUPONT
RUE:      11,RUE NOBEL
VILLE:    MONTIGNY

FLÈCHE HAUT POUR
ZONES ARRIÈRES
```

```
10 '------- SAISIE ECRAN
20 '
30 '---------------- NOMS DES ZONES
40 NRUB=3                ' NOMBRE DE RUBRIQUES
50 NRUB$(1)="NOM:":NRUB$(2)="RUE:":NRUB$(3)="VILLE:"
60 GOSUB 110
70 '------------- Pour test seulement
80 PRINT:FOR P=1 TO NRUB:PRINT FICH$(P):NEXT P
90 END
100 '=============== Sous-Programme saisie ecran ==
110 CLS
120 LOCATE 1,23:PRINT "FLECHE HAUT POUR ZONE ARRIERE"
130 '--------------- affichage noms des zones
140 FOR P=1 TO NRUB
150  LOCATE 1,P+1:PRINT NRUB$(P)
160  LOCATE 10,P+1:PRINT FICH$(P)    ' ancienne valeur
170 NEXT P
180 '------------- SAISIE DES ZONES
190 FOR P=1 TO NRUB
200  XL=10:YL=P+1:GOSUB 290
210  IF R=1 THEN FICH$(P)=LIG$
220  IF R=3 THEN IF P>1 THEN  LOCATE XL,YL:PRINT FICH$(P):P=
P-1:GOTO 200 ELSE 200
230 '
240  LOCATE XL,YL:PRINT FICH$(P);
250  X=15-LEN(FICH$(P)):IF X>0 THEN PRINT SPC(X)
260 NEXT P
270 RETURN
280 '------------------ Saisie d'une ligne
290 LIG$=""
300 '
310 L=LEN(LIG$):LOCATE XL+L,YL  ' coordonees affichage
320 '
330 C$=INPUT$(1):C=ASC(C$)      ' lecture 1 caractere
350 '
360 IF C<>8 AND C<>29 THEN 390            ' code suppression
370 IF L>0 THEN LIG$=LEFT$(LIG$,L-1):PRINT CHR$(8);CHR$(32);
:GOTO 310 ELSE 310
380 '
390 IF C=13 THEN 460            ' code de return
400 IF C=30 THEN R=3:RETURN     ' zone arriere
410 IF C<32 OR C>128 THEN BEEP:GOTO 310   ' controle
420 LIG$=LIG$+C$                ' ajout caractere
430 PRINT C$                    ' affichage caractere
440 GOTO 310
450 '--- return
460 IF LIG$<>"" THEN R=1 ELSE R=2
470 RETURN
```

HISTOGRAMME

Ce programme d'histogramme calcule l'échelle automatiquement.




```
10 '-------------- HISTOGRAMME
20 NM=7                    ' nombre de mois
30 CE=6:CF=15:COLOR CE,CF
40 SCREEN 2
50 OPEN "GRP:" FOR OUTPUT AS #1
60 DATA JV,500,   FV,400
70 DATA MRS,600,  AVR,700
80 DATA MAI,600,  JUI,500
90 DATA JL,700
100 '--
110 FOR I=1 TO NM:READ MOIS$(I),VNTE(I):NEXT I
120 '
130 XA=20:YA=150       ' depart axes
140 IX=30              ' intervalle
150 HECR=120           ' hauteur ecran
160 '------------- recherche maxi
170 MX=VNTE(1)
180 FOR M=2 TO NM
190  IF VNTE(M)>MX THEN MX=VNTE(M)
200 NEXT M
210 ECH=HECR/MX         ' echelle
220 '-------------------- axes
230 LINE(XA,YA)-(XA+NM*IX,YA),CE
240 LINE(XA,YA)-(XA,YA-HECR),CE
250 '------------------ AFFICHAGE MOIS
260 FOR M=1 TO NM
270  X=XA+8+IX*(M-1):Y=YA+10
280  DRAW "BM=X;,=Y;":PRINT #1,MOIS$(M)
290 NEXT M
300 '------------------ courbe
310 FOR M=1 TO NM
320  X1=IX*M:Y1=YA-VNTE(M)*ECH
```

→

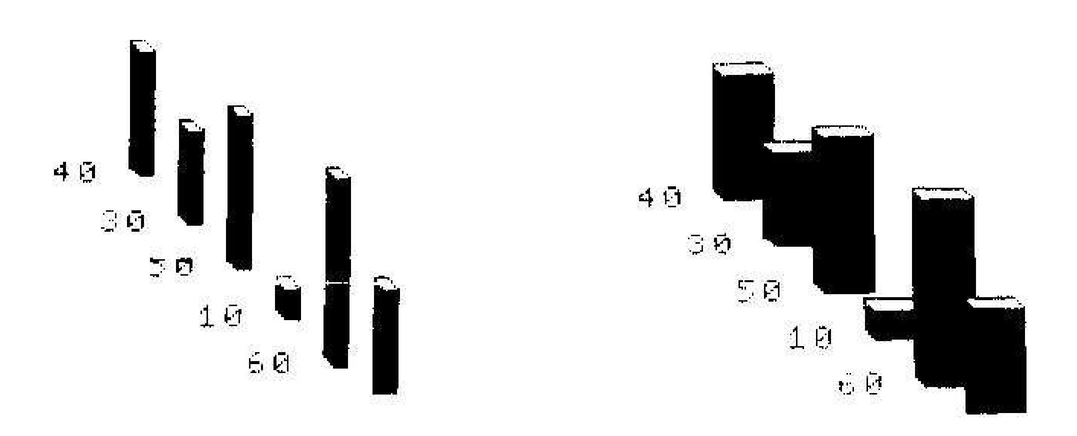
```
330  CE=M-INT(M/15)*14
340  LINE (X1,Y1)-(X1+8,YA),CE,BF
350 NEXT M
360 '------------------- affichage valeurs
370 FOR M=1 TO NM
380  Y=YA-VNTE(M)*ECH-16
390  X=XA+IX*(M-1)-8
400  DRAW "BM=X;,=Y;":PRINT #1,VNTE(M)
410 NEXT M
420 C$=INPUT$(1)
```

HISTOGRAMME EN 3 DIMENSIONS

Si la largeur des "batons" est augmentée, il faut supprimer les instructions 270 à 290 afin d'éviter des mélanges de couleurs. Une autre solution consiste à définir la même couleur pour tous les batons (instructions 260 et 290).

```
10 '-------- histogramme 3d
20 '
30 OPEN "GRP:" FOR OUTPUT AS #1
40 COLOR 1,15,15:SCREEN 2
50 XD=50:YD=60          ' depart
60 '
70 LG=4                 ' largeur
80 PR=4                 ' profondeur
90 ITV=15               ' intervalle
100 '
110 NH=6                ' nombre de batons
120 '-
130 H(1)=40
140 H(2)=30
150 H(3)=50
160 H(4)=10
170 H(5)=60
180 H(6)=40
190 '---
200 FOR H=1 TO NH            ' NH batons.
210  XB=XD+(H-1)*ITV
220  YB=YD+(H-1)*ITV
230  PRESET(XB-30,YB)
240  PRINT #1,H(H)
250  FOR DX=1 TO PR          ' 1 baton
260   LINE(XB+DX,YB+DX)-(XB+DX+LG,YB+DX-H(H)),H,BF
265  '---- 270 a 290 optionnel
270   IF DX=1 THEN 300
280   LINE (XB+DX,YB+DX-H(H))-(XB+DX+LG,YB+DX-H(H)),15
290   PSET(XB+DX+LG,YB+DX-H(H)),H
300  NEXT DX
310 NEXT H
320 C$=INPUT$(1)
```

→



BIBLIOTHÈQUE

Une bibliothèque est gérée dans des tables en mémoire centrale.
Ces tables sont sauvegardées sur cassette.

TITR$( )	AUT$( )	C$(,)	
MA VIE ET LA PSYCHANALYSE	FREUD	PSYCHANALYSE	
PROGRAMMER EN ASSEMBLEUR	PINAUD	ASSEMBLEUR	Z80
Titres	Auteur		

Deux mot-clés par ouvrage sont prévus. L'édition des listes triées par titre et par mot-clé se fait selon le principe présenté dans le programme de gestion d'adresses.
On remarquera l'instruction :

360 IF TIT$=LEFT$(TITR$(RANG,L)) THEN 480

En "mode modification", elle permet d'entrer seulement les premières lettres du titre.

```
10 '   BIBLIOTHEQUE
20 '
30 CLEAR 5000
40 DIM TITR$(100),AUT$(100),C$(100,2)
50 DIM CLE$(100),IX%(100)
60 CLS
70 NFICH=0                          ' nombres de fiches
80 R$="":INPUT "NOUVEAU FICHIER (O/N) ";R$
90 IF R$="O" THEN 120
100 GOSUB 1070
110 '
120 CLS
130 PRINT "MODES:":PRINT
140 PRINT TAB(2);"C      : CREATION/MODIFICATION"
150 PRINT TAB(2);"LTITRE: LISTE TRIEE PAR TITRE"
160 PRINT TAB(2);"LCLE   : LISTE PAR MOT CLE"
170 PRINT TAB(2);"FIN    : SAUVEGARDE CASSETTE" →
```

```
180 PRINT TAB(2);"S      : SUPPRESSION OUVRAGE"
190 PRINT:M$="":INPUT "MODE  )";M$
200 IF M$="C" THEN GOSUB 290
210 IF M$="LTITRE" THEN GOSUB 660
220 IF M$="FIN" THEN GOSUB 940
230 IF M$="LCLE" THEN GOSUB 1320
240 IF M$="S" THEN GOSUB 1530
250 GOTO 130
260 '================ CREATION/ MODIFICATION ====
270 '  En mode modification,entrer les premieres lettres du
     titre
280 '
290 PRINT
300 PRINT:TIT$="":INPUT "TITRE(ou return) ";TIT$
310 IF TIT$="" THEN RETURN
320 '
330 L=LEN(TIT$)
340 IF NFICH=0 THEN 400
350 FOR RANG=1 TO NFICH
360  IF TIT$=LEFT$(TITR$(RANG),L) THEN 480
370 NEXT RANG
380 '
390 '----------- nouveau titre
400 R$="":INPUT "Nouveau titre (O/N) ";R$
410 IF R$<>"O" THEN 300
420 NFICH=NFICH+1
430 RANG=NFICH
440 '
450 TITR$(RANG)=TIT$
460 '---
470 'Pour modification,appuyer sur <return> si une zone ne
     change pas
480 PRINT:PRINT TITR$(RANG):PRINT
490 PRINT AUT$(RANG);TAB(15);                ' ancienne valeur
500 AUT$="":INPUT "Auteur ";AUT$:IF AUT$<>"" THEN AUT$(RANG)=
    AUT$
510 '
520 FOR C=1 TO 2
530  PRINT C$(RANG,C);TAB(15);      ' ancienne valeur
540  C$="":INPUT "Mot cle ";C$
550  IF C$<>"" THEN C$(RANG,C)=C$
560 NEXT C
570 GOTO 300
```

```
650 '================== LISTE TRIEE PAR TITRE
660 FOR LV=1 TO NFICH
670  CLE$(LV)=TITR$(LV):IX%(LV)=LV
680 NEXT LV
690 '
700 NCL=NFICH:GOSUB 820
710 '---------- EDITION
720 PRINT:PRINT "LISTE PAR TITRE":PRINT
730 IF NCL=0 THEN RETURN
740 FOR LV=1 TO NCL
750  X=IX%(LV)
760  PRINT LEFT$(TITR$(X),20);TAB(25);
770  PRINT AUT$(X)
780 NEXT LV
790 PRINT:INPUT "APPUYER SUR <RETURN> ";R$
800 RETURN
```
→

```
810 '------------- TRI SHELL
820 ECART=NCL
830 '
840 ECART=INT(ECART/2):IF ECART<1 THEN RETURN
850 IV=0
860 FOR I=1 TO NCL-ECART
870  J=I+ECART
880  IF CLE$(J)=>CLE$(I) THEN 910
890  SWAP CLE$(I),CLE$(J)
900  SWAP IX%(I),IX%(J):IV=1
910 NEXT I
920 IF IV=1 THEN 850 ELSE 840
930 '=========================== SAUVEGARDE CASSETTE
940 PRINT "APPUYEZ SUR <RECORD> PUIS <RETURN>
950 C$=INPUT$(1)
960 OPEN "BIB" FOR OUTPUT AS #1
970 '
980 FOR I=1 TO NFICH
990  PRINT #1,TITR$(I)
1000  PRINT #1,AUT$(I)
1010  FOR C=1 TO 2
1020   PRINT #1,C$(I,C)
1030  NEXT C
1040 NEXT I
1050 CLOSE #1:RETURN
1060 '===================== LECTURE CASSETTE
1070 PRINT "APPUYEZ SUR <PLAY> ":PRINT
1080 OPEN "BIB" FOR INPUT AS #1
1090 FOR I=1 TO 100
1100  IF EOF(1)=-1 THEN CLOSE #1:NFICH=I-1:GOTO 1180
1110  INPUT #1,TITR$(I),AUT$(I)
1120  FOR C=1 TO 2
1130   INPUT #1,C$(I,C)
1140  NEXT C
1150 NEXT I
1160 STOP
1170 '
1180 PRINT:PRINT NFICH;"LIVRES":PRINT
1190 FOR TP=1 TO 1000:NEXT TP
1200 RETURN

1310 '============== LISTE PAR MOT CLE"
1320 PRINT
1330 NCL=0                     ' nombre de cles
1340 FOR LV=1 TO NFICH
1350  FOR C=1 TO 2
1360   IF C$(LV,C)="" OR C$(LV,C)="*" THEN 1380
1370   NCL=NCL+1:CLE$(NCL)=C$(LV,C):IX%(NCL)=LV
1380  NEXT C
1390 NEXT LV
1400 '
1410 GOSUB 820
1420 '----- edition
1430 PRINT:PRINT "LISTE PAR MOT-CLE":PRINT
1440 FOR LV=1 TO NCL
1450  X=IX%(LV)
1460  IF CLE$(LV-1)<>CLE$(LV) THEN PRINT :PRINT CLE$(LV):PRINT
1470  PRINT TAB(4);LEFT$(TITR$(X),20);TAB(25);
1480  PRINT AUT$(X)
1490 NEXT LV
```

→

```
1500 PRINT:INPUT "APPUYEZ SUR <RETURN>";R$
1510 RETURN
1520 '==================================== SUPPRESSION TITRE
1530 PRINT:TIT$="":INPUT "TITRE ";TIT$
1540 IF TIT$="" THEN RETURN
1550 L=LEN(TIT$)
1560 FOR RANG=1 TO NFICH
1570  IF TIT$=LEFT$(TITR$(RANG),L) THEN 1610
1580 NEXT RANG
1590 PRINT:PRINT "N'EXISTE PAS":PRINT:GOTO 1530
1600 '
1610 R$="":INPUT "SUPPRESSION OK (O/N) ";R$
1620 IF R$<>"O" THEN 1530
1630 FOR J=RANG TO NFICH-1
1640  SWAP TITR$(J),TITR$(J+1)
1650  SWAP AUT$(J),AUT$(J+1)
1660  FOR C=1 TO 2
1670   SWAP C$(J,C),C$(J+1,C)
1680  NEXT C
1690 NEXT J
1700 TITR$(NFICH)="":AUT$(NFICH)=""
1710 FOR C=1 TO 2:C$(NFICH,C)="":NEXT C
1720 NFICH=NFICH-1
1730 GOTO 1530
```

```
        C      : CREATION/MODIFICATION
        LTITRE: LISTE TRIEE PAR TITRE
        LCLE   : LISTE PAR MOT CLE
        FIN    : SAUVEGARDE CASSETTE
        S      : SUPPRESSION OUVRAGE

MODE   ?? LTITRE

LISTE PAR TITRE

BASIC DE A A Z              DUPONT
BASIC ET SES FICHIER        DUPONT
BASIC POUR TOUS             DUPONT
FORTRAN                     MARTIN
PROGRAMMER EN ASSEMB        PINAUD
VIPERE AU POING             BAZIN

ASSEMBLEUR

    PROGRAMMER EN ASSEMB PINAUD

BASIC

    BASIC DE A A Z       DUPONT
    BASIC ET SES FICHIER DUPONT
    BASIC POUR TOUS      DUPONT

FICHIER

    BASIC ET SES FICHIER DUPONT
    BASIC DE A A Z       DUPONT

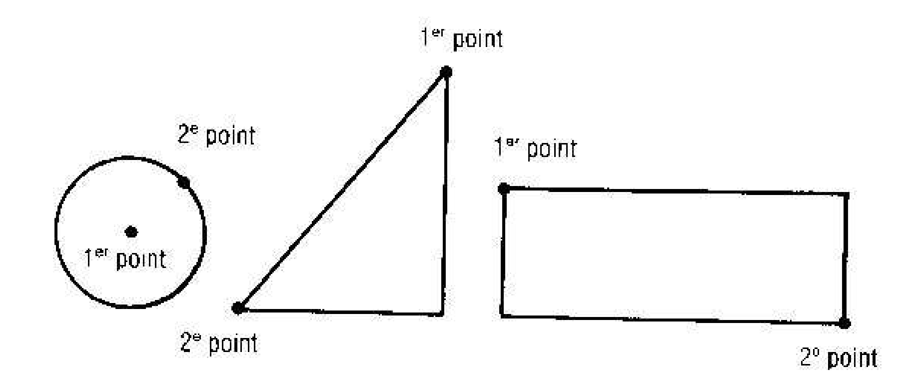
FORTRAN

    FORTRAN              MARTIN
```

POSSIBILITÉS GRAPHIQUES DU MSX

DESSIN

A l'aide d'un curseur que vous déplacez avec les quatre flèches, vous réalisez un dessin que vous pouvez sauvegarder.
Trois types de figures sont prévus : **cercle, rectangle, triangle.**
Il faut deux points pour définir chaque type de figure. La couleur de fond(F) permet d'effacer des parties de figure et ainsi d'en composer d'autres.

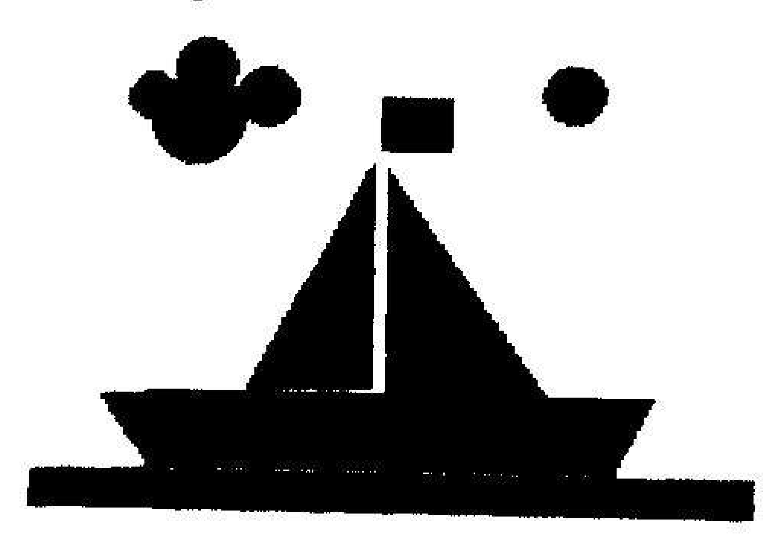


La couleur peut être modifiée en mode curseur en frappant un chiffre compris entre 1 et 9.

Exemple, pour tracer un rectangle :

- Vous répondez "R" à la question "Commande? (R,T,C,)"
- Vous positionnez le curseur avec les flèches puis vous appuyez sur "V".
- Vous déplacez le curseur et vous appuyez sur "V".

Remarque : en couleur de fond(F), le curseur disparaît lorsque vous quittez une zone coloriée.
Dans ce cas, il faut changer la couleur.

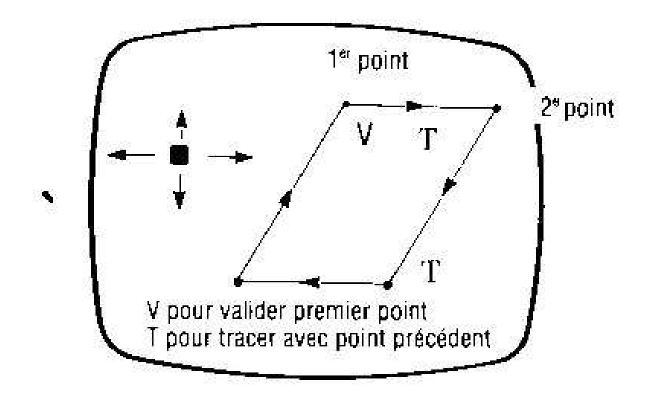


```
10 '------------------------ DESSIN
20 ' (CURSEUR SPRITE)
30 CE=1:CF=15              ' ecriture/fond
40 COLOR CE,CF:SCREEN 2
50 OPEN "GRP:" FOR OUTPUT AS #1
60 X=100:Y=100:SPRITE$(1)=CHR$(191)   '     curseur
70 '
80 PRESET(14,150):PRINT #1,"CLAVIER MAJUSCULE"
85 '
90 COLOR 1:PRESET (14,160):PRINT #1,"Commande? (R,C,T)  "
100 CM$=INPUT$(1)
110 P=INSTR("RTC",CM$):IF P=0 THEN BEEP:GOTO 90
120 '
130 PRESET(14,170):PRINT #1,"1ER POINT":GOSUB 370
140 XA=X:YA=Y
150 COLOR 1:PRESET(94,170):PRINT #1,"2EME POINT":GOSUB 370
160 XB=X:YB=Y
170 IF CM$="R" THEN LINE(XA,YA)-(X,Y),CE,BF
180 IF CM$="T" THEN GOSUB 290
190 IF CM$="C" THEN GOSUB240
200 COLOR CF:PRESET(14,170):PRINT #1,STRING$(30,CHR$(200))
210 COLOR CE
220 GOTO 90
230 '--------------- CERCLE
240 R=SQR((XA-X)^2+(YA-Y)^2)
250 IF R<2 THEN RETURN
260 CIRCLE (XA,YA),R,CE:PAINT(XA,YA+2),CE
270 RETURN
280 '------------ TRIANGLE
290 IF YB-YA=0 THEN RETURN
300 P=(XB-XA)/(YB-YA)
310 FOR Y=YA TO YB STEP SGN(YB-YA)
320  IF (Y-YA)*P=0 THEN 340
330  LINE(XA,Y)-(XA+(Y-YA)*P,Y),CE
340 NEXT Y
350 RETURN
360 '------------------GESTION  CURSEUR
370 PRESET(14,180):PRINT #1,"FLECHES PUIS "V" / COOL(1-9),F"
380 PUT SPRITE 1,(X,Y-1),CE,1
390 '
400 C$=INKEY$:IF C$="" THEN 400
410 '
420 C=ASC(C$)
430 IF C=29 THEN IF X>2 THEN X=X-2
440 IF C=28 THEN IF X<250 THENX=X+2
450 IF C=30 THEN IF Y>2 THEN Y=Y-2
460 IF C=31 THEN IF Y<189 THEN Y=Y+2
470 '
480 IF C$="V" THEN PSET(X,Y),CE:T=CE:GOTO 530
490 IF VAL(C$)>0 AND VAL(C$)<9 THEN CE=VAL(C$)
500 IF C$="F" THEN CE=CF
510 GOTO 380
520 '
530 COLOR CF:PRESET(14,180):PRINT #1,STRING$(31,CHR$(200))
540 COLOR CE
550 RETURN
```

TRACÉ D'UN DESSIN PAR SEGMENTS DE DROITES ET DIGITALISATION D'UN DESSIN

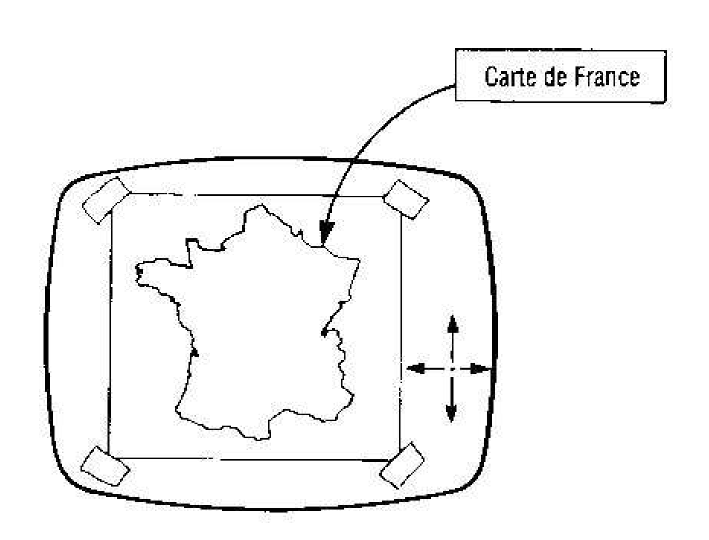
Le programme ci-dessous permet de déplacer un curseur à l'aide de quatre flèches →← ↑ ↓, de "valider" des points et de tracer des droites entre ces points.

- **Pour "valider" un point,** nous appuyons sur "V".
- **Pour tracer une droite** entre le point courant et le point valide précédent, nous appuyons sur "T".



Le programme permet également de "digitaliser" un dessin :

- Nous collons sur l'écran une feuille transparente sur laquelle a été décalquée la carte de France par exemple.
- Nous déplaçons le curseur sur le périmètre de la figure et périodiquement, nous "validons" les points. Les coordonnées X et Y des points s'affichent alors à l'écran.



```
10 '------------ TRACE DE DESSIN PAR DROITES +DIGITALISATION
20 '
30 CF=15:CE=1                   ' couleur fond et ecriture
40 COLOR CE,CF:SCREEN 2
50 '
60 OPEN "GRP:" FOR OUTPUT AS #1
70 PSET(14,155):PRINT #1,"CLAVIER MAJUSCULE"
80 PSET(14,165):PRINT #1,"1ER POINT:FLECHES PUIS 'V'"
90 PSET(14,175)
100 PRINT #1,"AUTRES POINTS:FLECHES PUIS:"
110 PSET(14,185):PRINT #1,"T:TRACE DROITE"
120 X=100:Y=100               ' Coordonnees depart
130 SPRITE$(1)=CHR$(191)
140 '------------------- curseur (sprite)
150 PUT SPRITE 1,(X,Y-1),CE,1
160 '
170 C$=INKEY$:IF C$="" THEN 170 ' test clavier
180 '
190 C=ASC(C$)
200 IF C=29 THEN X=X-1         ' gauche
210 IF C=28 THEN X=X+1         ' droite
220 IF C=31 THEN Y=Y+1         ' bas
230 IF C=30 THEN Y=Y-1         ' haut
240 IF C$="V" THEN PSET(X,Y),CE:XA=X:YA=Y:GOSUB 280
250 IF C$="T" THEN LINE(XA,YA)-(X,Y),CE:XA=X:YA=Y:GOSUB 280
260 GOTO 150
270 '----------------------- affichage X,Y
280 COLOR CF:PSET(12,145)
290 FOR I=1 TO 10:PRINT #1,CHR$(200);:NEXT I 'effacement
300 COLOR CE
310 PSET (12,145):PRINT #1,X;Y
320 RETURN
330 '----
340 ' ex: Frapper 'V' puis deplacer curseur
350 '     et frapper 'T'
```

TRACÉ DE DESSIN DÉFINI EN DATA

Le programme ci-dessous trace une figure en reliant entre eux des points définis en DATA.

```
10 '------------ CARTE DE FRANCE
20 COLOR 1,15
30 SCREEN 2
40 DATA 135,13
50 DATA 140,19,   151,28,   160,33
60 DATA 167,36,   177,37,   174,45
70 DATA 171,56,  170,68,   166,75
80 DATA 165,80,  168,82,   172,80
90 DATA 171,90,  173,98,   180,107
100 DATA 180,112, 167,121,  158,116
110 DATA 152,118,  148,121,  146,127
120 DATA 144,135,  125,129,  106,123
130 DATA 111,100,  111,85,  105,77
```

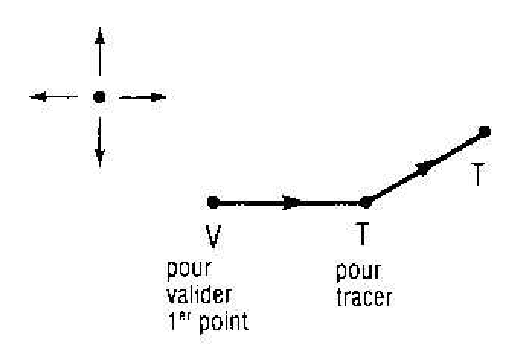
→

```
140 DATA 100,64,    94,57,    86,54
150 DATA 86,46,    97,44,    106,48
160 DATA 103,39, 102,30,    107,30
170 DATA 111,36, 116,37,    121,31
180 DATA 128,23,  130,16,  134,14
190 DATA 999,999
200 '---
210 READ XA,YA          ' Premier Point
220 PSET (XA,YA),1
230 '--
240 READ X,Y:IF X=999 THEN 280
250 '
260 LINE-(X,Y),1
270 GOTO 240
280 C$=INPUT$(1)
```

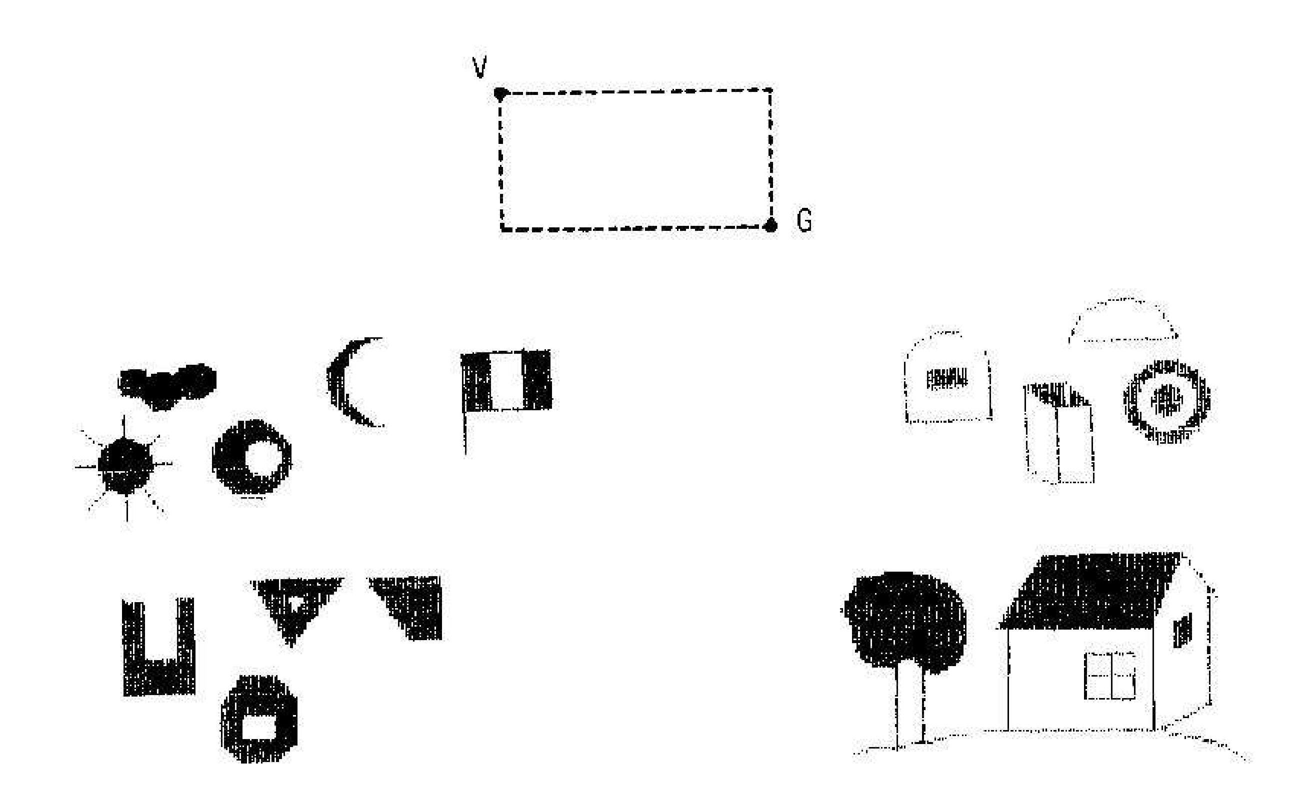
DESSINATEUR

Permet de dessiner des pôlygones par segments de droites.

- Appuyez sur "V" pour valider le premier point.
- Déplacez le curseur avec les quatre flèches puis appuyez sur "T" pour tracer une droite avec le point précédent. Pour une figure discontinue, utilisez "V".



- Pour tracer un cercle, appuyez sur "V" pour le centre, puis positionnez le curseur sur un point de la circonférence et appuyez sur "C".
- "P" peint une figure fermée (positionnez le curseur à l'intérieur de la figure).
- "A" annule le dernier tracé (pour une droite).
- 1,2,3... détermine la couleur d'écriture.
- "F" permet de dessiner en couleur de fond sur une partie d'écran coloriée. Dans ce cas, le curseur disparaît sur une partie d'écran non peinte (changez la couleur pour faire apparaître le curseur).
- "G" permet de gommer toute la surface d'un rectangle.



Les commandes pourraient être stockées dans une table.
Ainsi, les dessins pourraient être sauvegardés et modifiés en ajoutant ou en supprimant des commandes.

On remarquera la gestion du curseur avec un sprite.

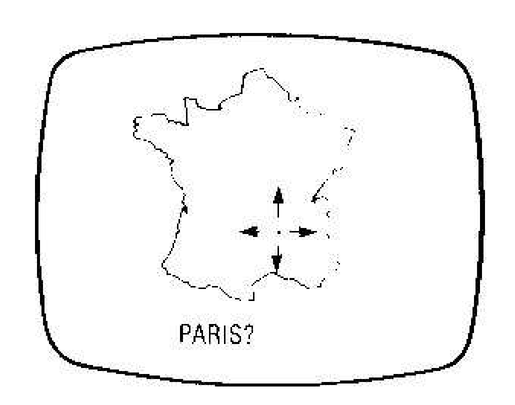
```
10 '--------- DESSINATEUR
20 CF=15:CE=1                    ' couleur fond et ecriture
30 COLOR CE,CF
40 SCREEN 2
50 '
60 OPEN "GRP:" FOR OUTPUT AS #1
70 PRESET(18,135):PRINT #1,"CLAVIER MAJUSCULE"
80 PRESET(18,145):PRINT #1,"1ER POINT:FLECHES PUIS 'V'"
90 PRESET(18,155):PRINT #1,"2EME POINT:FLECHES PUIS:"
100 PRESET(18,165):PRINT #1,"T:TRACE DROITE/ C:CERCLE"
110 PRESET (18,175):PRINT #1,"A:ANNULATION TRACE / P:PEINDRE"
120 PRESET(18,185):PRINT #1,"1,2,3,.F COULEURS/ G:GOMMER"
130 X=100:Y=100                  ' Coordonnees depart
140 XA=X:YA=Y                    ' coordonnees point precedent
150 XB=XA:YB=YA
160 SPRITE$(1)=CHR$(191)          ' curseur
170 '--------------- curseur (sprite)
180 PUT SPRITE 1,(X,Y-1),CE,1
190 '
200 C$=INKEY$:IF C$="" THEN 200  ' test clavier
210 '
220 C=ASC(C$)
230 IF C=29 THEN X=X-1:GOTO 180 ' gauche
240 IF C=28 THEN X=X+1:GOTO 180 ' droite
250 IF C=31 THEN Y=Y+1:GOTO 180 ' bas
260 IF C=30 THEN Y=Y-1:GOTO 180 ' haut
```

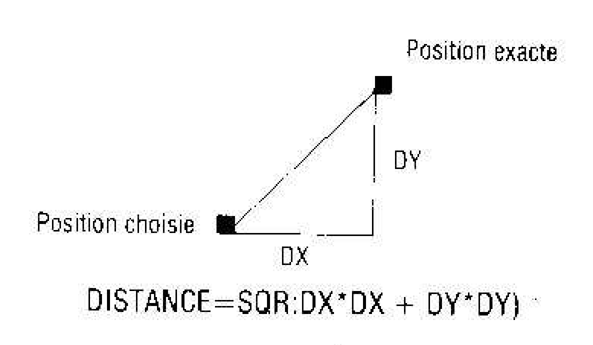
→

```
270 IF C>32 THEN GOSUB 380
280 IF C$="V" THEN PSET(X,Y),CE:XA=X:YA=Y
290 IF C$="T" THEN LINE(XA,YA)-(X,Y),CE:XB=XA:YB=YA XA=X:YA=Y
300 IF C$="C" THEN R=SQR((X-XA)^2+(Y-YA)^2):CIRCLE(XA,YA),R,CE
310 IF C$="G" THEN LINE(XA,YA)-(X,Y),CF,BF
320 IF C$="A" THEN LINE(XB,YB)-(X,Y),CF:X=XB:Y=YB:XA=X:YA=Y
330 IF C$="P" THEN PAINT (X,Y),CE
340 IF VAL(C$)<>0 THEN CE=VAL(C$)
350 IF C$="F" THEN CE=CF
360 GOTO 180
370 '--
380 PRESET(18,125):COLOR CF:PRINT #1,CHR$(200)
390 PRESET(18,125):COLOR CE:PRINT #1,C$
400 RETURN
```

INTERROGATION DE GÉOGRAPHIE

Le programme ci-après permet de tester les connaissances en géographie. Nous demandons de situer la position d'une ville. L'élève déplace un curseur et valide la position choisie par "V". Nous donnons à l'élève l'écart entre la position choisie et la position exacte.





On pourra remplacer la gestion du curseur par un SPRITE.

- Supprimer : 700,730,740,750,770
- Ajouter : 530, SPRITE$(1)=CHR$(191)
  700 PUT SPRITE 1,(X,Y−1),1,1
  720 C$=INKEY$:IF C$="" THEN 720

```
10 '--------------- GEOGRAPHIE
20 CLEAR 1000
30 DIM XV(20),YV(20),V$(20)
40 COLOR 1,15:SCREEN 2
50 '------------- CARTE DE FRANCE
60 DATA 135,13
70 DATA 140,19,   151,28,   160,33
80 DATA 167,36,   177,37,   174,45
90 DATA 171,56,  170,68,   166,75
100 DATA 165,80,  168,82,   172,80
110 DATA 171,90,  173,98,   180,107
120 DATA 180,112, 167,121,  158,116
```

→

```
130 DATA 152,118,  148,121,  146,127
140 DATA 144,135,  125,129,  106,123
150 DATA 111,100,  111,85,  105,77
160 DATA 100,64,   94,57,   86,54
170 DATA 86,46,   97,44,   106,48
180 DATA 103,39, 102,30,   107,30
190 DATA 111,36, 116,37,   121,31
200 DATA 128,23,  130,16,  134,14
210 DATA 999,999
220 '--------------- VILLES
230 DATA 137,25,LILLE
240 DATA 147,42,REIMS
250 DATA 134,49,PARIS
260 DATA 120,56,MANS
270 DATA 104,56,RENNES
280 DATA 88,51,BREST
290 DATA 114,100,BORDEAUX
300 DATA 153,91,LYON
310 DATA 155,109,AVIGNON
320 DATA 162,116,MARSEILLE
330 DATA 174,113,NICE
340 DATA 130,115,TOULOUSE
350 DATA 140,128,PERPIGNAN
360 DATA 999,999,ZZZ
370 '---
380 READ XA,YA        ' Premier Point
390 PSET (XA,YA),1
400 '--
410 READ X,Y:IF X=999 THEN 460
420 '
430 LINE-(X,Y),1
440 GOTO 410
450 '--
460 V=0
470 READ X,Y,V$:IF X=999 THEN NV=V:GOTO 520
480 V=V+1
490 XV(V)=X:YV(V)=Y:V$(V)=V$
500 GOTO 470
510 '---
520 OPEN "GRP:" FOR OUTPUT AS #1
530 '
540 '----------
550 X=130:Y=90
560 '
570 V=INT(RND(1)*NV)+1        ' ville au hasard
580 '
590 PRESET(10,160):PRINT #1,"OU EST SITUE:";V$(V)

670 '------ GESTION CURSEUR
680 PSET(10,170):PRINT #1,"FLECHES PUIS 'V' "
690 '
700 T=POINT(X,Y)
710 '
720 C$=INKEY$:IF C$<>"" THEN 770
730 PSET (X,Y),1
740 PRESET(X,Y)
750 GOTO 720
760 '
770 PSET(X,Y),T
780 '
```

OU EST SITUE:LYON

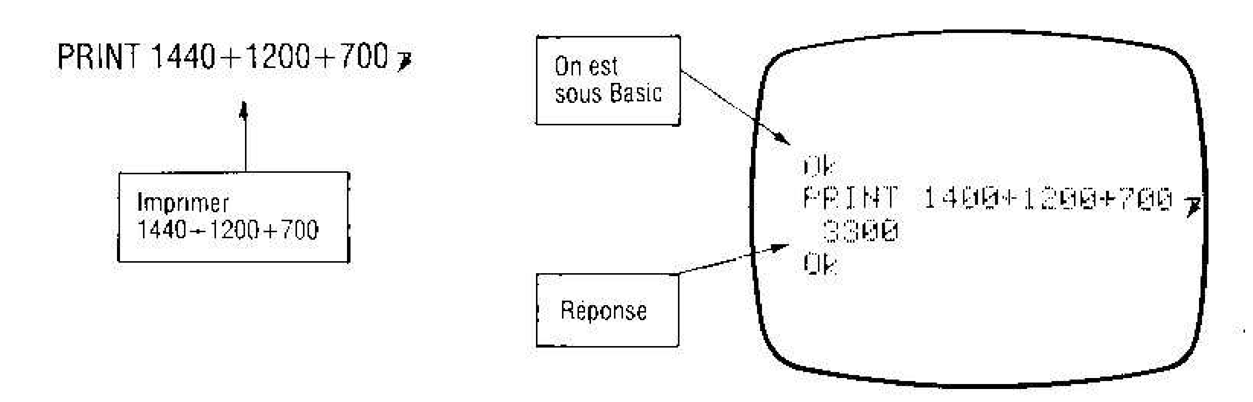
```
790 C=ASC(C$)
800 IF C=29 THEN X=X-2
810 IF C=28 THEN X=X+2
820 IF C=30 THEN Y=Y-2
830 IF C=31 THEN Y=Y+2
840 '
850 IF C$="V" OR C$="v" THEN 890
870 GOTO 700
880 '-------------- calcul distance
890 DX=XV(V)-X:DY=YV(V)-Y
900 D=SQR(DX*DX+DY*DY)
910 PSET(10,180):PRINT #1,"VOUS ETES A:";D*10;"KM"
920 PSET(XV(V),YV(V)),1
930 FOR TP=1 TO 2000:NEXT TP
940 '--------- effacement--------
950 PRESET(XV(V),YV(V))
960 COLOR 15:PSET(10,180):PRINT #1,STRING$(31,CHR$(200))
970 PSET(10,160):PRINT #1,STRING$(30,CHR$(200))
980 COLOR 1
990 GOTO 570
```

ANNEXE | 1

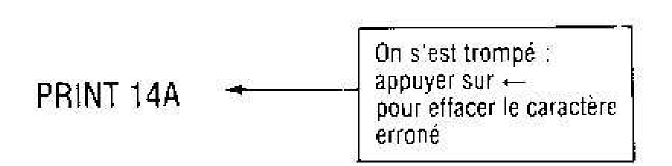
INITIATION

LE MODE DIRECT

Sans écrire de programme, on peut utiliser directement les instructions du BASIC. Par exemple, pour imprimer la somme de 1400, 1200 et 700, nous écrivons :

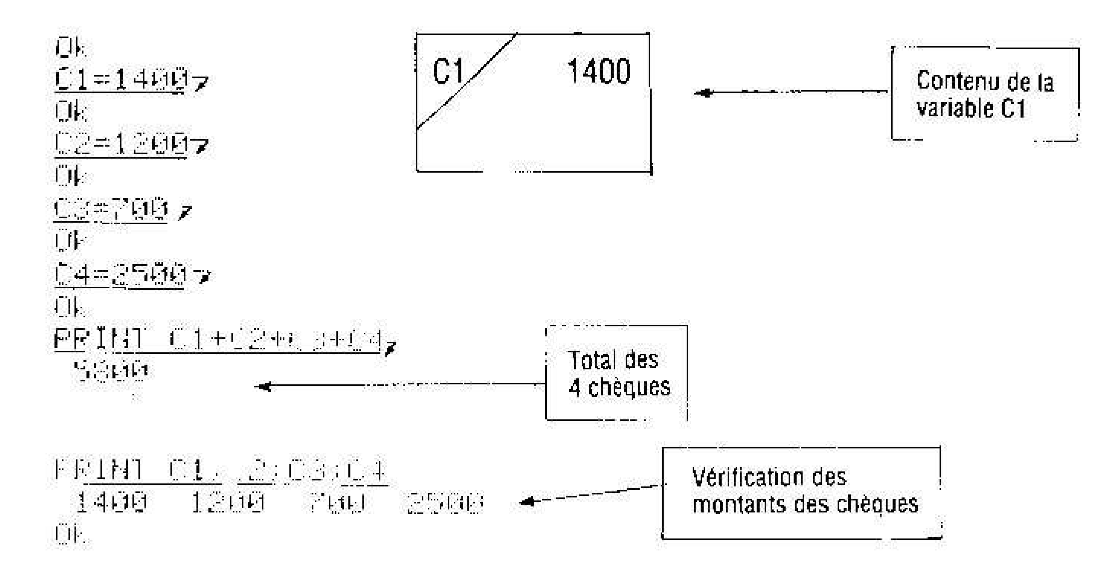


Le symbole ↗ correspond à la touche "RETURN" qui valide la ligne frappée au clavier. Tant que nous n'avons pas appuyé sur cette touche, les caractères de la ligne en cours peuvent être annulés en appuyant sur "←".



LES VARIABLES

Les montants de quatre chèques, dont nous voulons calculer le total, peuvent être mémorisés, grâce à des "variables" :
En frappant C1=1400 nous affectons, à la variable C1, la valeur 1400.



Si le montant d'un chèque est faux, rien n'empêche de lui affecter une autre valeur :

```
C3=800
Ok
PRINT C1+C2+C3+C4
 5900
Ok
```

Attention ! Les noms des variables peuvent comporter plusieurs lettres mais, seules les 2 premières lettres sont significatives.

```
CH1=1400
Ok
CH2=1200
Ok
PRINT CH1,CH2
 1200          1200
```

CH1=CH2

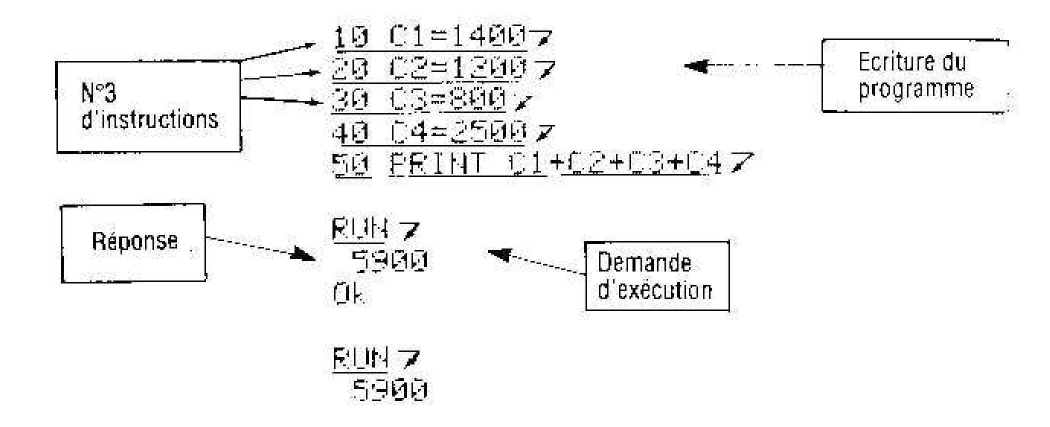
Les noms de variables ne doivent pas comporter de mot-clé du Basic :

"TO" est un mot-clé du Basic

```
Ok
TOTAL=100
Syntax error
```

LE MODE PROGRAMME

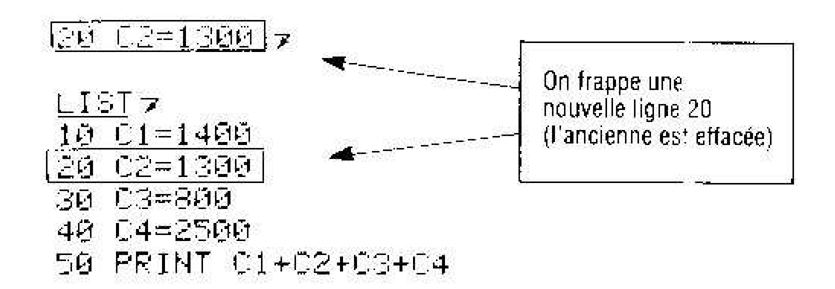
Si une même suite d'instructions doit être exécutée plusieurs fois, il devient plus pratique d'écrire un programme qui sera exécutable autant de fois que nous le désirons :
- □ chaque instruction est précédée d'un numéro,
- □ "RUN" commande l'exécution du programme,
- □ les instructions sont exécutées dans l'ordre de la numérotation.



A l'issue de l'exécution, les variables ont conservé leurs valeurs, et il est toujours possible de faire en "MODE DIRECT" (c'est-à-dire sans numéro d'instruction).



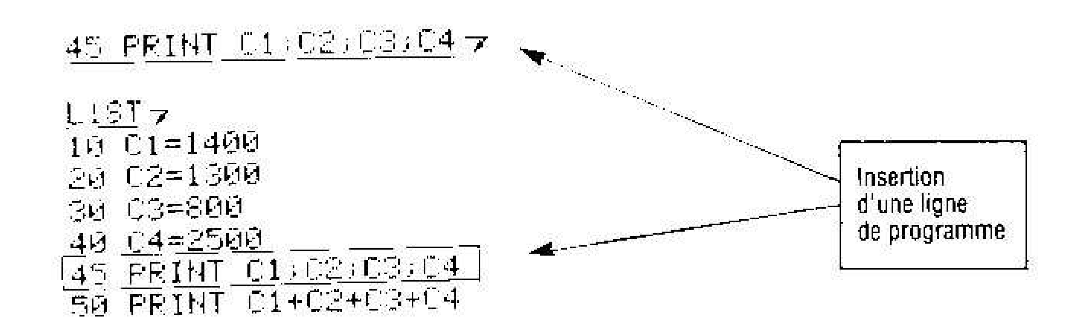
Si une instruction est mal frappée, la frapper à nouveau.



La commande LIST permet de lister un programme.

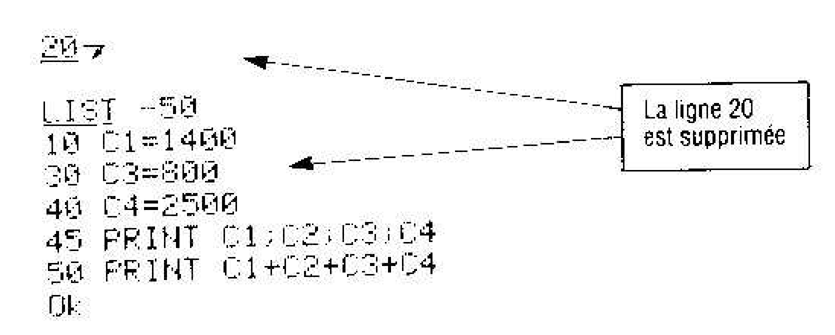
■ Ajout d'une ligne de programme

Vous pouvez également insérer des instructions en leur affectant un numéro compris entre ceux des lignes où vous voulez insérer :

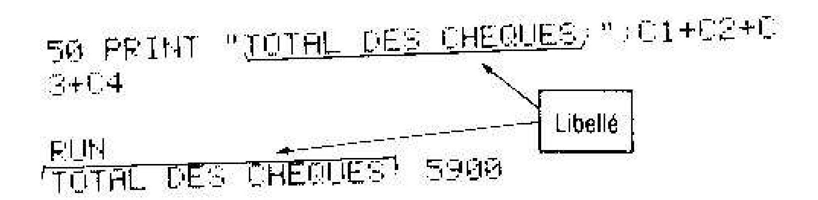


■ Suppression d'une ligne de programme

Frapper le numéro de ligne à supprimer, suivi de "RETURN" :



La présentation du résultat pourrait être améliorée :



■ Sauvegarde du programme

La sauvegarde sur cassette d'un programme se fait par la commande :

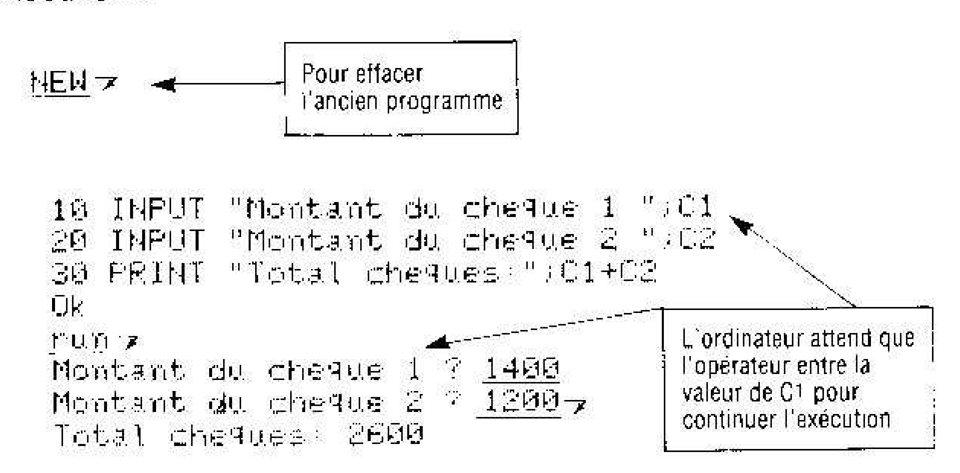
CSAVE "nom-programme"

La touche "RECORD" du lecteur/enregistreur doit être enfoncée.
La commande **CLOAD "nom-programme"** permet de relire un programme sauvegardé.

CSAVE "ESSAI"
CLOAD "ESSAI"

L'INSTRUCTION INPUT

Cette instruction permet de fournir des informations au programme pendant son exécution :



Au moment où une instruction est exécutée, le message qui y figure (montant chèque 1? sur l'exemple) est affiché, puis l'ordinateur attend la réponse de l'opérateur avant de poursuivre l'exécution du programme.

Remarque : La commande NEW permet d'effacer un programme.

Si, en réponse à une question INPUT, l'opérateur appuie sur "RETURN" sans entrer de valeur, la variable spécifiée dans INPUT conserve son ancienne valeur. Naturellement, il convient d'en tenir compte dans l'écriture des programmes.

```
10 INPUT "NOMBRE ";X
20 PRINT X
30 INPUT "NOMBRE ";X
40 PRINT X
Ok
RUN
NOMBRE ? 12
 12
NOMBRE ?
 12
```

X a conservé son ancienne valeur

LES BOUCLES

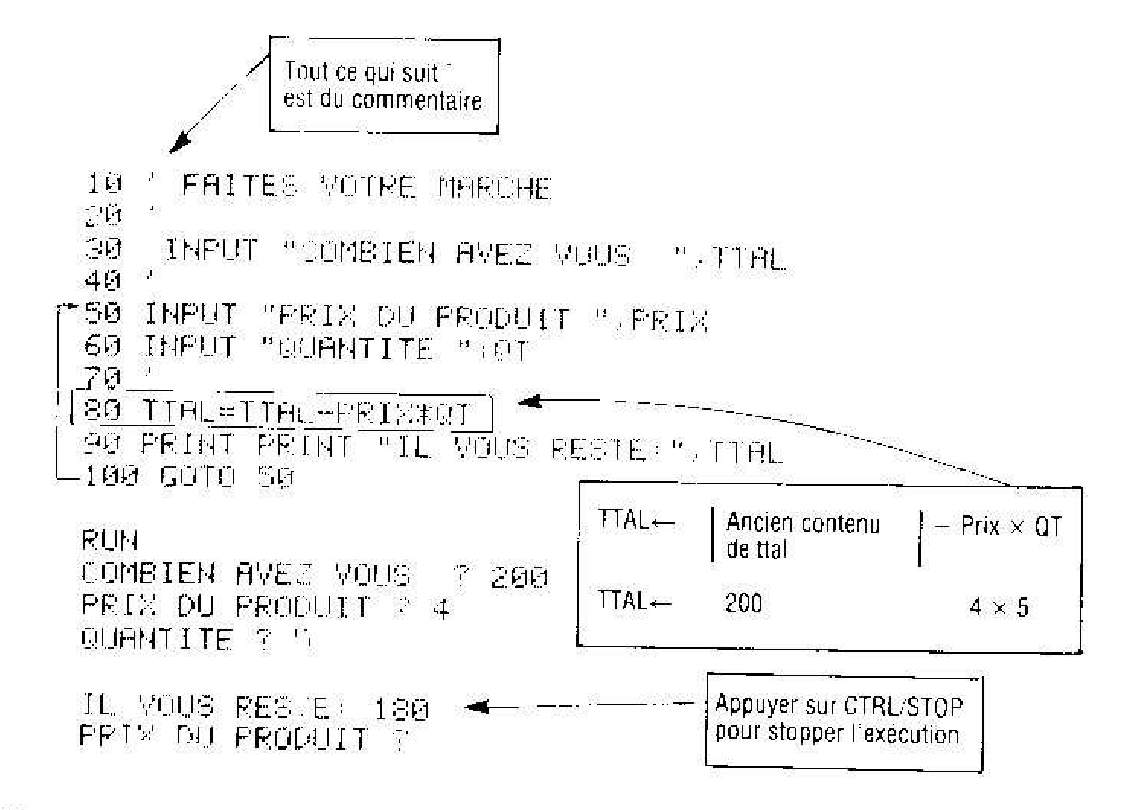
Voici un programme très court et néanmoins bavard. L'instruction GOTO 10 (ALLER EN 10) provoque un "branchement" à l'instruction 10 qui est à nouveau exécutée.

Boucle

```
10 PRINT "APPUYER SUR CTRL ET STOP"
20 GOTO 10

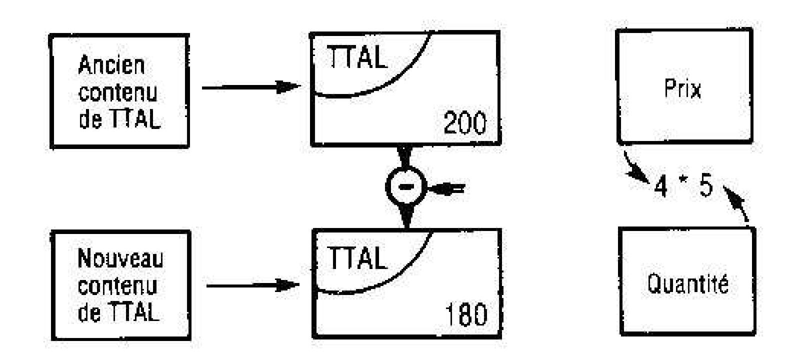
RUN
APPUYER SUR CTRL ET STOP
APPUYER SUR CTRL ET STOP
APPUYER SUR CTRL ET STOP
APPUYER SUR CTRL ET STOP
```

Le programme ci-dessous vous permet de connaître de quelle somme vous disposez au fur et à mesure de vos achats.



Tout ce qui suit **REM** ou le caractère ' (apostrophe) est du commentaire qui n'est pas exécuté par BASIC.
Par exemple, la ligne 10 n'est pas exécutée.

L'instruction 80 ne doit pas être lue comme une égalité algébrique.
Le signe "=" représente une affectation :



L'ordinateur soustrait du total (TTAL) le prix multiplié par la quantité (PRIX*QTE) puis range le résultat dans TTAL.

Remarques sur l'instruction PRINT :

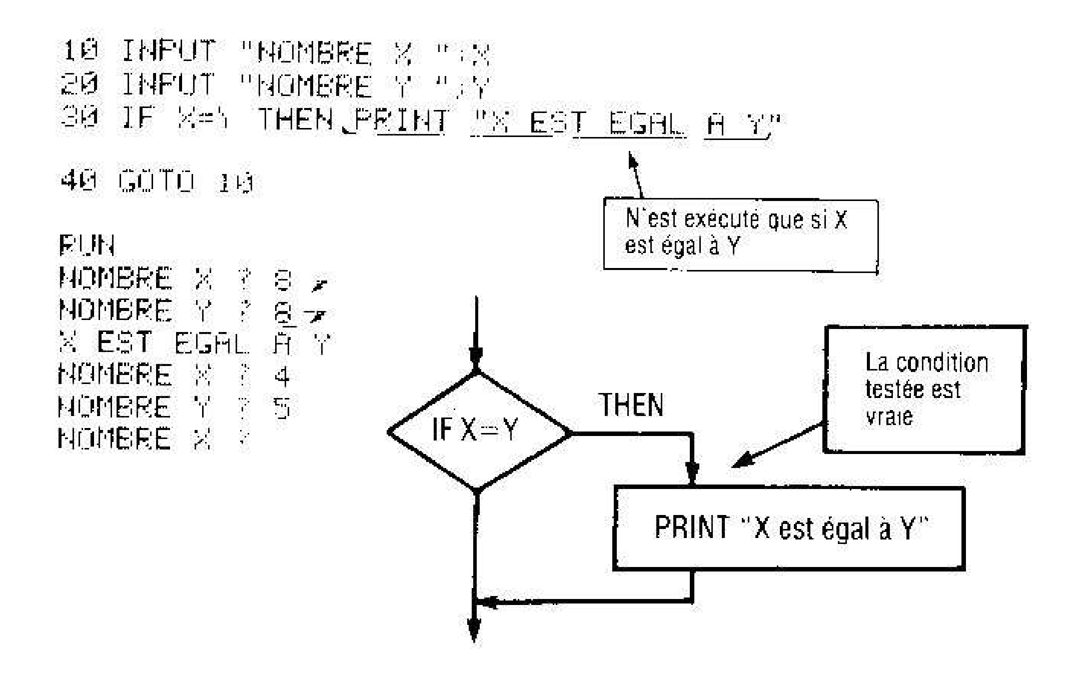
Un point-virgule empêche le saut de ligne.

```
10 X=123
20 Y=456
30 PRINT X;
40 PRINT Y;
Ok
run
 123  456
```

La présence d'un point-virgule après PRINT X provoque l'affichage de "456" à la suite de "123".

IF condition THEN instruction (SI... ALORS)

L'instruction IF...THEN.. teste si une condition est vraie et, dans ce cas, exécute une ou plusieurs instructions (séparées par ":").
L'instruction 30 se lit : SI X=Y ALORS IMPRIMER "X est égal à Y"



Les différentes opérateurs de comparaison sont :

égalité : =
plus grand que : >
plus petit que : <
différent de : <>

Le programme ci-dessous présente 2 nombres à l'opérateur et lui demande quel en est le produit.
Si la réponse est fausse, la question est à nouveau posée.

```
10 A=9
20 B=8
30 PRINT "Produit de ";A;"Par";B;
40 INPUT R
50 IF R=A*B THEN PRINT "C'est ca":STOP
60 PRINT "Erreur"
70 GOTO 30
 Ok
 run
 Produit de  9 Par 8 ? 72
 C'est ca
 Break in 50
```

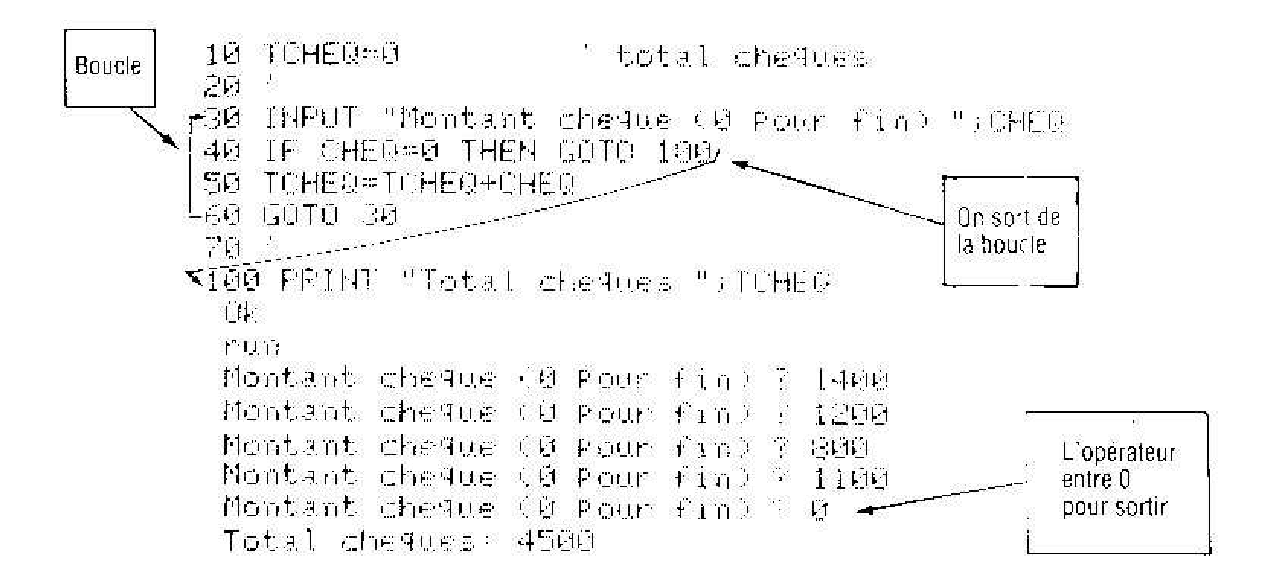
L'instruction 50 doit se lire :

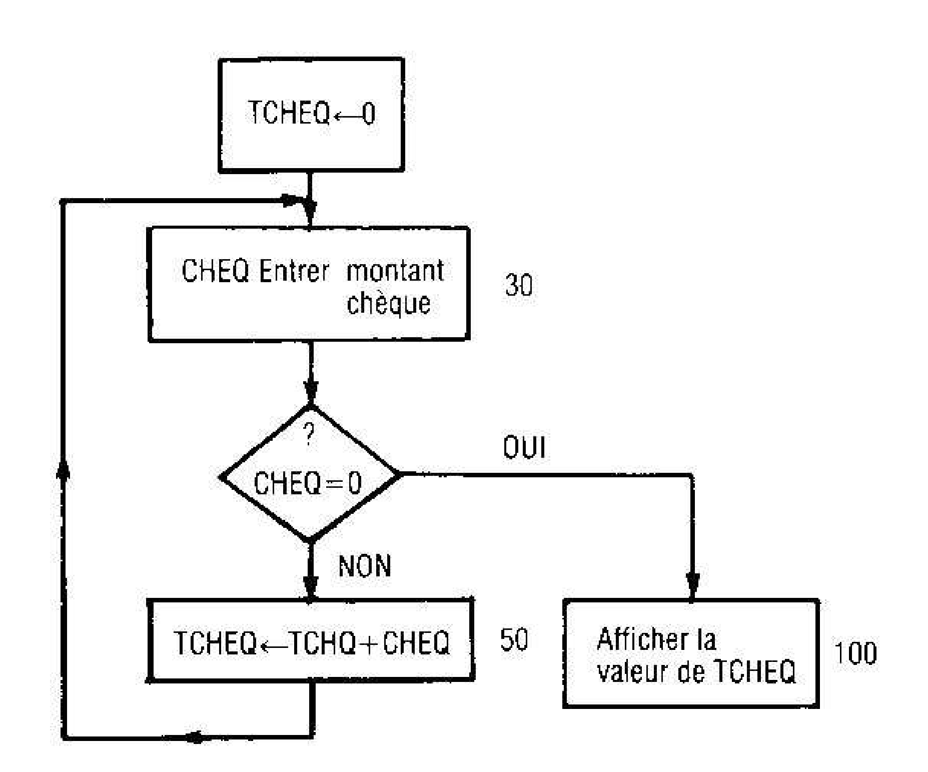
SI R=A*B ALORS IMPRIMER "Oui, c'est ça"

Sans instruction de test, ne pourraient être exécutées que des séquences d'instructions figées. C'est dans cette instruction IF...THEN... que réside toute la puissance de l'ordinateur.

■ Sortie de boucle

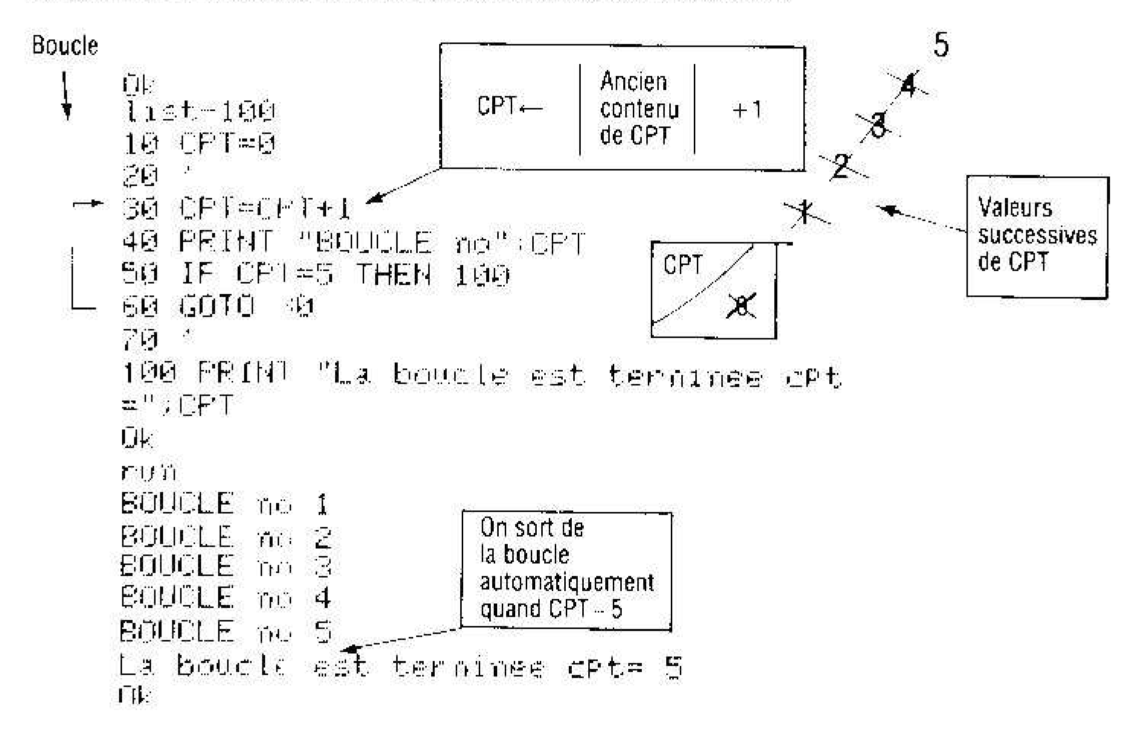
Pour effectuer la somme de plusieurs chèques, au lieu d'écrire une suite d'instructions INPUT, utilisons une **boucle.** Lorsqu'il n'y a plus de chèques à totaliser, l'opérateur entre la valeur 0 en réponse à la question "Montant cheque?".


```
10 TCHEQ=0              ' total cheques
20 '
30 INPUT "Montant cheque (0 pour fin) ";CHEQ
40 IF CHEQ=0 THEN GOTO 100
50 TCHEQ=TCHEQ+CHEQ
60 GOTO 30
70 '
100 PRINT "Total cheques ";TCHEQ
Ok
run
Montant cheque (0 pour fin) ? 1400
Montant cheque (0 pour fin) ? 1200
Montant cheque (0 pour fin) ? 800
Montant cheque (0 pour fin) ? 1100
Montant cheque (0 pour fin) ? 0
Total cheques  4500
```

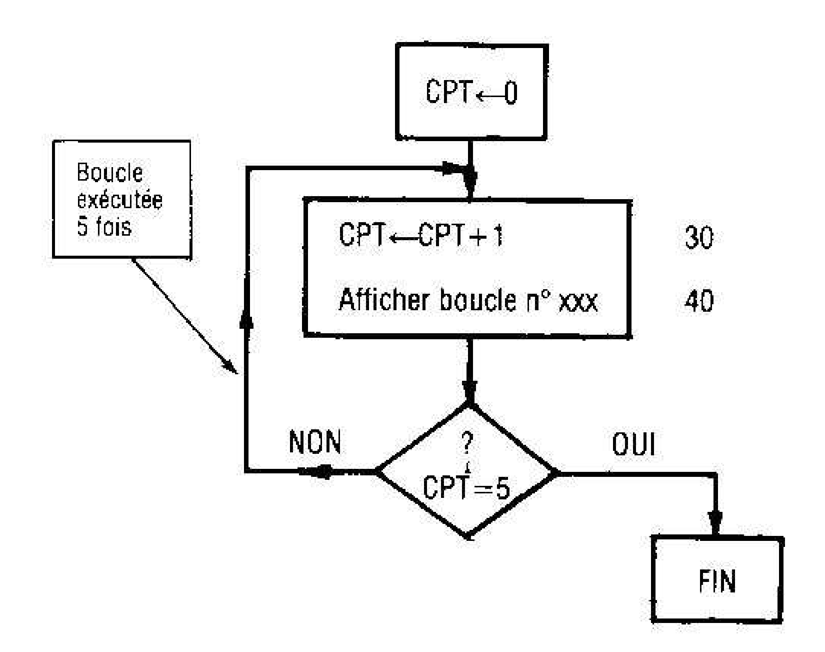



■ Compteur de boucles

Introduisons maintenant une autre notion, celle de "compteur de boucles". Afin d'exécuter une boucle un certain nombre de fois seulement, nous augmentons une "variable compteur" de 1 à chaque passage dans la boucle, puis nous testons la valeur de cette variable compteur : si le contenu de ce compteur a atteint la valeur limite souhaitée, nous sortons de la boucle.

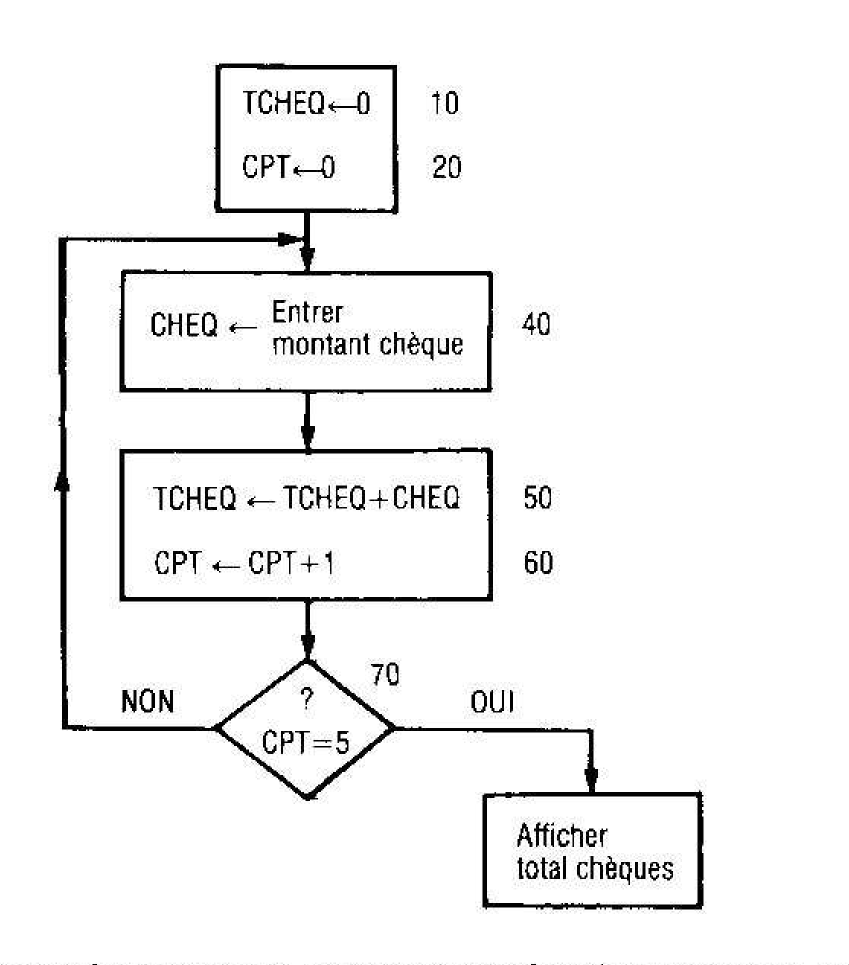


La boucle est terminée CPT=5 ;



Utilisons un compteur de boucles pour notre totalisation de chèques. Nous quittons la boucle automatiquement dès que 5 chèques ont été totalisés.

```
10 TCHEQ=0                        ' total cheques
20 CPT=0                          ' compteur de boucles
30 '
40 INPUT "Montant cheque ";CHEQ
50 TCHEQ=TCHEQ+CHEQ               ' cumul cheque
60 CPT=CPT+1                      ' augmenter CPT de 1
70 IF CPT=5 THEN GOTO 100         ' SI CPT=5 ALORS ALLER EN 100
80 GOTO 40
90 '
100 PRINT "Total des cheques:";TCHEQ
Ok
run
Montant cheque ? 1400
Montant cheque ? 1200
Montant cheque ? 800
Montant cheque ? 1100
Montant cheque ? 500
Total des cheques: 5000
```



Question : Comment faire pour que ce programme fonctionne avec un nombre de chèques non prévu à l'avance ?

Réponse : 5 INPUT "Combien de chèques ?"; NCH
70 IF CPT=NCH THEN GOTO 100

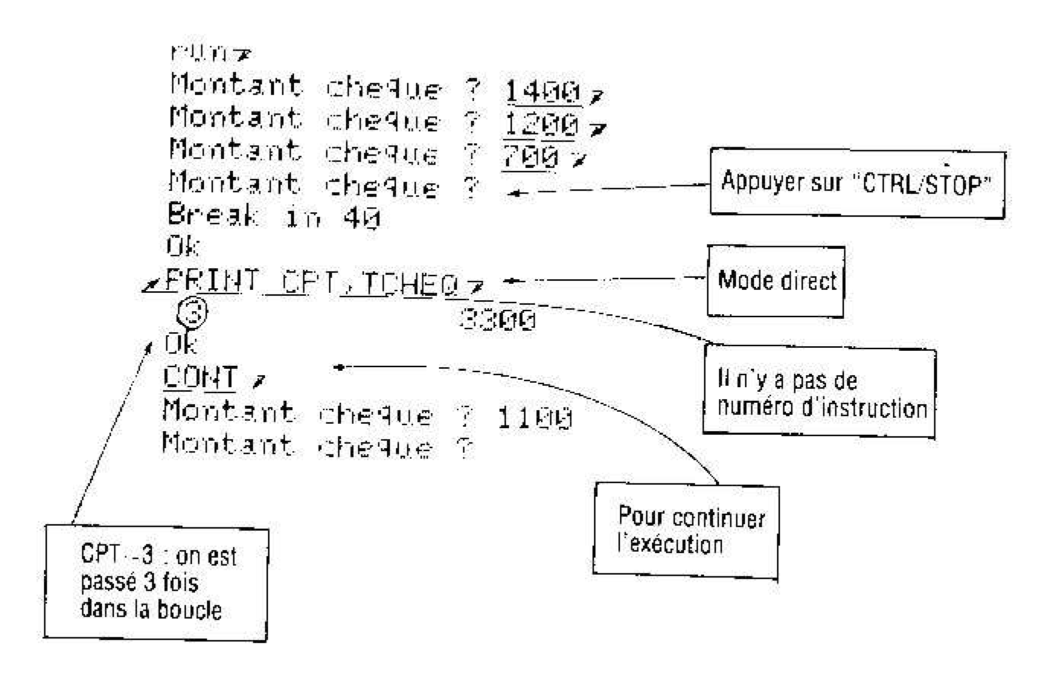
Pour vous assurer que vous avez bien compris ce programme :

- entrer les montants pour 3 chèques :
- au lieu d'entrer le montant pour le quatrième chèque, appuyez sur -CTRL/STOP.
- frappez en mode direct :

PRINT CPT,TC

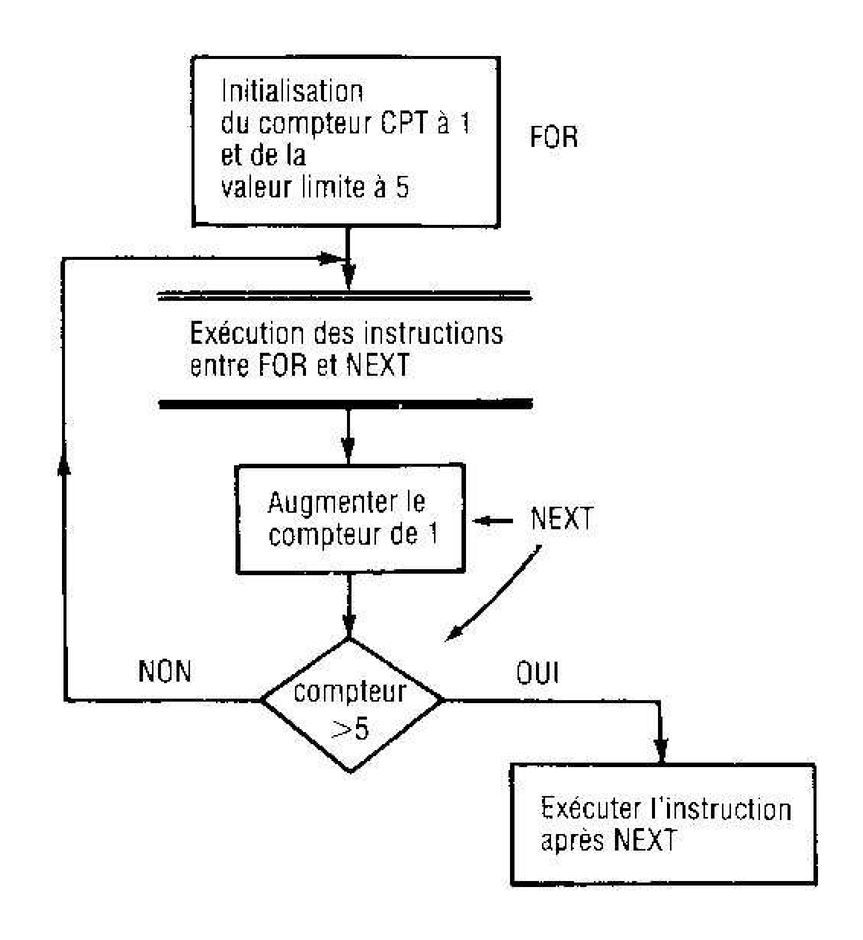
CPT doit être égal à 3 et TCHEQ doit avoir, pour valeur, le total des trois chèques déjà entrés.

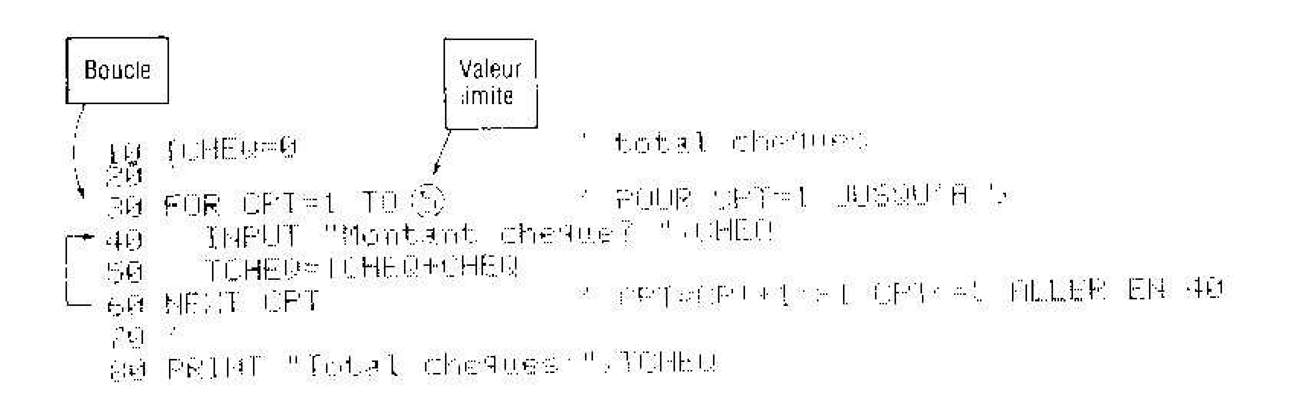
- frappez "CONT" pour continuer l'exécution du programme.



LA BOUCLE FOR

Le programme précédent pourrait s'écrire plus simplement à l'aide d'une boucle FOR...NEXT.. :





Comment se déroule l'exécution de ce programme :

- au moment où l'instruction **FOR** est exécutée, le système BASIC affecte à CPT la valeur 1 et enregistre la valeur limite spécifiée (5 sur l'exemple) ;
- les instructions entre **FOR** et **NEXT** sont exécutées avec CPT=1.

L'instruction NEXT CPT augmente la valeur de CPT de un.
Si celle-ci reste inférieure ou égale à la valeur limite spécifiée dans **FOR**, les instructions entre **FOR** et **NEXT** sont à nouveau exécutées avec CPT=2, etc.

Autres exemples :

```
10 FOR I=1 TO 100
20      PRINT "Le BASIC, mais c'est très simple"
30 NEXT I

RUN

Le BASIC, mais c'est très simple
Le BASIC, mais c'est très simple
```

Ce programme imprime les carrés des nombres de 1 à 5.

```
10 FOR I=1 TO 5
20   PRINT I,I*I
30 NEXT I
Ok
run
 1              1
 2              4
 3              9
 4              16
 5              25
Ok
```

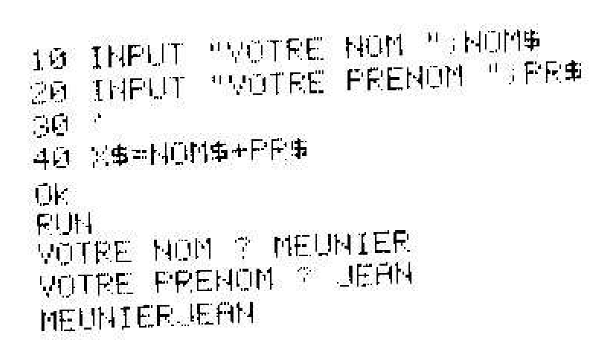
LES CHAÎNES DE CARACTÈRES

Les variables que nous avons jusqu'à présent considérées, étaient du type numérique. Il existe également des variables du type "**chaîne de caractères**". Pour les distinguer, elles comportent, à la fin de nom, le caractère "$".

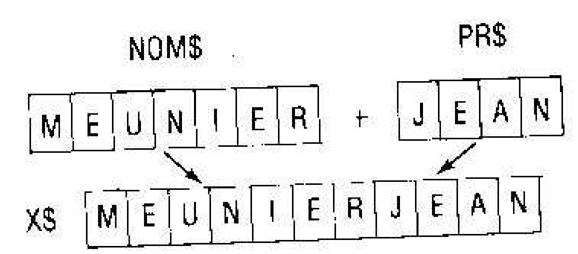
Exemple :

```
10 INPUT "Quel est votre nom ";NOM$
20 PRINT NOM$;
30 GOTO 20
Ok
RUN
Quel est votre nom ? DUPONT
DUPONTDUPONTDUPONTDUPONTDUPONTDUPONTD
UPONTDUPONTDUPONTDUPONTDUPONTDUPONTDU
PONTDUPONTDUPONTDUPONTDUPONTDUPONTDUP
```

Le ";" après NOM$ empêche le saut de ligne. Ainsi, le nom est affiché plusieurs fois sur la même ligne. Toutefois, en fin de ligne, il y a saut automatique à la ligne suivante.
Nous verrons, plus tard, qu'il existe des instructions de manipulation des chaînes de caractères.
La concaténation de chaînes se fait à l'aide de l'opérateur "+".



```
10 INPUT "VOTRE NOM ";NOM$
20 INPUT "VOTRE PRENOM ";PR$
30 ?
40 X$=NOM$+PR$
OK
RUN
VOTRE NOM ? MEUNIER
VOTRE PRENOM ? JEAN
MEUNIERJEAN
```



ANNEXE | 2

MESSAGES D'ERREURS DU BASIC

Si le message d'erreur ne suffit pas pour détecter d'où provient l'erreur, on pensera à visualiser les valeurs des variables en "mode direct". Ceci aidera bien souvent à la retrouver.
Des instructions STOP judicieusement placées permettront également de mieux suivre l'évolution des valeurs des variables. On pourra aussi insérer momentanément des instructions de visualisation des valeurs de variables (PRINT) "X-"; X par exemple).
La trace (avec TRON), un peu trop riche en informations, ne sera utilisée que dans les cas particulièrement délicats.

(* signale les messages spécifiques au disque)

Message	Code	
***Bad allocation table**	60	La table d'allocation de la disquette est détruite.
***Bad drive name**	62	Mauvais nom d'unité. Utiliser A:,B:,,..
***Bad file mode**	61	(mauvaise utilisation de fichier) On utilise PUT,GET avec un fichier ouvert en séquentiel où on ouvre un fichier dans un autre mode que "R" "O" ou "I" ou "A".
***Bad file number**	52	Mauvais numéro de fichier.
***Bad file name**	56	(mauvais nom de fichier) Le nom de fichier utilisé n'est pas normalisé (trop de caractères par exemple).

Message	Code	
Can't continue	17	(l'exécution ne peut se poursuivre) On a tenté de poursuivre l'exécution d'un programme qui : ☐ a été stoppé à cause d'une erreur ☐ a été modifié après une interruption ☐ n'existe pas. On peut cependant continuer par GOTO xx (RUN xx initialise les variables à 0)
Direct statement in file	57	Commande directe dans un fichier.
Disk full	66	(disque plein) Tout l'espace disque est utilisé.
Disk I/O error	69	(erreur disque) Une erreur disque s'est produite pendant une lecture/écriture. Le système d'exploitation ne peut rien faire.
***Write protected**	68	Le disque est protégé en écriture.
Division by zero	11	(division par zéro)
***Field overflow**	50	(dépassement du field) Une instruction FIELD a tenté d'allouer dans le buffer plus de caractères qu'il a été prévu à l'ouverture du fichier.
***File already exist**	65	(fichier déjà existant) Le nom de fichier défini dans NAME existe déjà.
File already open	54	(fichier déjà ouvert) On essaie d'ouvrir en mode séquentiel OUTPUT un fichier déjà ouvert ou un KILL tente de supprimer un fichier ouvert.
***File not found**	53	(fichier non trouvé) Une instruction LOAD,KILL ou OPEN référence un fichier qui n'existe pas.
***File still open**	64	Un fichier n'a pas été fermé par (CLOSE).
For without next	26	(FOR sans NEXT) Un FOR sans NEXT a été détecté.
Illegal direct	12	(illégal en mode direct) L'instruction frappée n'est pas valide en mode direct, elle ne peut être exécutée que précédée d'un numéro de ligne. Ex. : instruction INPUT-DEF FN

Message	Code	
Illegal function call	5	(appel illégal de fonction) Un paramètre hors du domaine normal a été passé à une fonction arithmétique ou une fonction chaîne. Ex. : □ argument négatif pour SQR □ longueur spécifiée dans LEFT$,MID$,RIGHT$ non comprise entre 0 et 255.
Input past end	55	(dépassement de fin de fichier) Une instruction INPUT#.. est exécutée alors qu'il n'y a plus d'information à lire (fichier vide éventuellement). Pour éviter cette erreur, utiliser la détection de fin de fichier EOF.
Internal error	51	(erreur interne) Une erreur système s'est produite.
Line buffer overflow	23	(dépassement du buffer de ligne) On essaie d'entrer une ligne avec trop de caractères.
Missing operand	24	(opérande manquant) Une expression contient un opérateur sans opérande.
NEXT without FOR	1	(NEXT sans FOR) Une variable dans un NEXT ne correspond à aucun FOR. Ex. : La ligne FOR correspondante a été effacée sans le NEXT associé.
No resume	19	(pas d'instruction RESUME) Une routine de traitement d'erreur qui ne contient pas d'instruction RESUME a été appelée.
Out of data	4	(Data épuisés) Un READ est exécuté alors qu'il n'y a plus de DATA à lire. □ On a oublié des DATA. □ On a oublié de programmer RESTORE
Out of memory	7	(plus de mémoire centrale) Il n'y a plus assez de place en mémoire centrale. □ On doit supprimer une partie du programme ou effacer des tableaux (ERASE).

Message	Code	
Out of string space	14	(plus de place pour les chaînes) L'espace pour les chaînes n'est pas suffisant (voir CLEAR).
Overflow	6	(dépassement de capacité) Ex. : On a tenté de donner à une variable entière une valeur en dehors de −32768,+32767.
Redimensionned array	10	(tableau redimensionné) Un tableau est à nouveau dimensionné ou une dimension de tableau est déclarée pour un tableau non déclaré explicitement mais créé par BASIC (avec une dimension 10) parce qu'il a déjà été référencé (par une instruction A(4)=X par exemple).
***Resume without error**	22	(RESUME sans erreur) Une instruction RESUME a été exécutée alors qu'il n'y avait pas d'erreur.
RETURN without GOSUB	3	(RETURN sans GOSUB) □ On est "entré" dans un sous-programme par GOTO (au lieu de GOSUB). □ On est "entré" dans un sous-programme par erreur parce que l'on a oublié STOP ou END à la fin de l'exécution de son programme (devant un sous-programme).
***Sequentiel I/O only**	58	On essaie de lire un fichier séquentiel en accès direct.
String formula too complexe	16	(expression chaîne trop complexe) Une expression du type chaîne est trop longue ou trop complexe. La décomposer en expressions plus courtes.
String too long	15	(chaîne trop longue) Une chaîne ne peut dépasser 255 caractères.
Subscript out of range	9	(référence en dehors du domaine) Un tableau est référencé en dehors de ses dimensions. Souvent, pour un tableau non déclaré qui a été dimensionné à 10 par BASIC parce que référencé (par une instruction A(4)=X par exemple).
Syntax error	2	(erreur de syntaxe) La ligne contient une erreur de syntaxe : □ parenthèses non appairées □ ponctuation incorrecte □ instruction n'existant pas □ etc.

Message	Code	
Type mismatch	13	(désaccord entre numérique et chaîne) Une valeur numérique est affectée à une chaîne où l'inverse.
***Too many files**	67	Il y a plus de fichiers que le nombre prévu par MAXFILES.
Undefined line	8	(numéro de ligne indéfini) Une instruction référence une ligne qui n'existe pas.
Undefined user fonction	18	(fonction utilisateur indéfinie) Une fonction USR est appelée avant d'être définie.
Unpritable error	23	(il n'y a pas de message pour cette erreur).

ANNEXE | 3

CARACTÈRES DE CONTRÔLE

Les caractères dont les codes sont compris entre 0 et 31 ont des fonctions particulières. A partir du clavier, on y accède avec la touche **CTRL** ou directement pour certains codes (← → ↑ ↓).
Dans un programme, on y accède par "**PRINT CHR$ (CODE)**".

1	CTRL/A	
2	CTRL/B	— Curseur au début du mot précédent.
3	CTRL/C	—
4	CTRL/D	—
5	CTRL/E	— Efface la ligne à droite du curseur.
6	CTRL/F	— Curseur au début du mot suivant.
7	CTRL/G	— Sonnerie.
8	CTRL/H	— Curseur arrière.
9	CTRL/I	— [TAB]
10	CTRL/J	— Saut de ligne
11	CTRL/K	— HOME
12	CTRL/L	— [CLS]
13	CTRL/M	— Retour en début de ligne.
14	CTRL/N	— Positionne le curseur en fin de ligne.
15	CTRL/O	—
16	CTRL/P	—
17	CTRL/Q	—
18	CTRL/R	— [INS]
19	CTRL/S	—
20	CTRL/T	—
21	CTRL/U	— Efface la ligne courante.
22	CTRL/V	—
23	CTRL/W	—
24	CTRL/X	— [SELECT]
25	CTRL/Y	—
26	CTRL/Z	—
27	CTRL/	—
28	CTRL/\	— Flèche droite
29	CTRL/]	— Flèche gauche
30	CTRL/↑	— Flèche haut
31	CTRL/-	— Flèche bas

— [DEL]

ANNEXE | 4

TABLE DES CODES ASCII

32 → 128

32		64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	¦
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	
64	@	96	`	128	

128 → 255

128	Ç	170	¬	212	▪
129	ü	171	½	213	▀
130	é	172	¼	214	▄
131	â	173	¡	215	⁜
132	ä	174	«	216	Δ
133	à	175	»	217	♦
134	å	176	Ã	218	ø
135	ç	177	ã	219	█
136	ê	178	Ĩ	220	▀
137	ë	179	ĩ	221	▌
138	è	180	Õ	222	▐
139	ï	181	õ	223	▄
140	î	182	Ũ	224	α
141	ì	183	ũ	225	ß
142	Ä	184	Ĳ	226	Γ
143	Å	185	ĳ	227	π
144	É	186	‡a	228	Σ
145	æ	187	¨	229	σ
146	Æ	188	◊	230	µ
147	ô	189	‰	231	τ
148	ö	190	¶	232	Φ
149	ò	191	§	233	Θ
150	û	192	▄	234	Ω
151	ù	193	▚	235	δ
152	ÿ	194	█	236	∞
153	Ö	195	▀	237	φ
154	Ü	196	▪	238	∈
155	¢	197	█	239	∩
156	£	198	▌	240	≡
157	¥	199	▞	241	±
158	₧	200	█	242	≥
159	ƒ	201	▕	243	≤
160	á	202	█	244	⌠
161	í	203	▨	245	⌡
162	ó	204	▧	246	÷
163	ú	205	▼	247	≈
164	ñ	206	▲	248	°
165	Ñ	207	▶	249	•
166	ª	208	◀	250	·
167	º	209	⧗	251	√
168	¿	210	⧓	252	ⁿ
169	⌐	211	●	253	²
170	¬	212	▪	254	■

CHR$(1) + CHR$(65+X)

65	☺
66	☻
67	♥
68	♦
69	♣
70	♠
71	·
72	◘
73	○
74	◙
75	♂
76	♀
77	♪
78	♫
79	☼
80	†
81	┴
82	┬
83	┤
84	├
85	┼
86	│
87	─
88	┌
89	┐
90	└
91	┘
92	╳
93	╱
94	╲
95	+
96	[illegible]

ANNEXE | 5

CARACTÈRES SPECIAUX

CODE

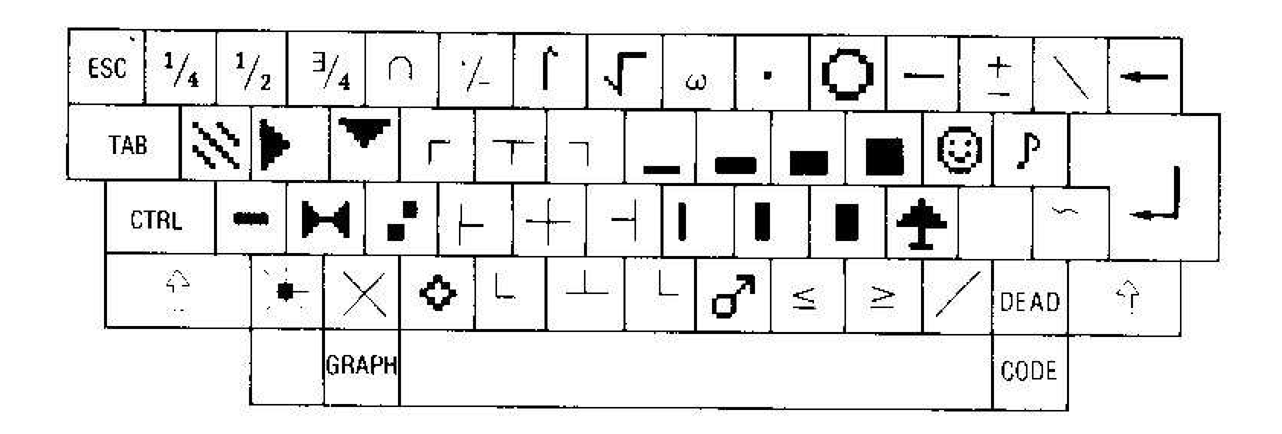
En appuyant sur la touche CODE, on obtient les caractères suivants :

ESC	ƒ	‡	§	ς	ÿ	α	β	γ	ç	δ	ε	θ	`	←
TAB	ȧ	è	ì	ô	û	á	é	í	ó	ú	φ	ω	↵	
CTRL	ä	ë	ï	ö	ü	ã	æ	ĩ	õ	ũ	ÿ	σ		
⇧	à	è	ì	ò	ù	ñ	µ	à	a̲	o̲	⇧			
	GRAPH		CODE											

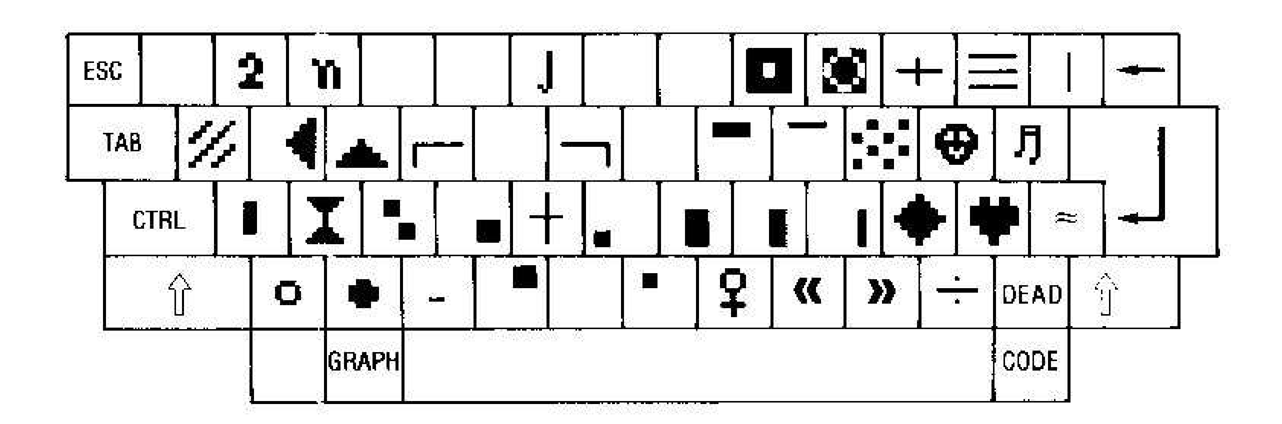
SHIFT CODE

ESC	¡	Pt	¶	£	¥			Γ	Ç	Δ				←
TAB							É			Π	Φ	Ω	↵	
CTRL	Ä			Ö	Ü	Ã	Æ	Ĩ	Õ	Ũ	Ŋ	Σ		
⇧						Ñ		À		¿	⇧			
↑	GRAPH		CODE											

GRAPH



SHIFT GRAPH



Achevé d'imprimer
sur les presses de l'imprimerie IBP
à Rungis (Val-de-Marne 94) 686.73.54
Dépôt légal - janvier 1985

N° d'impression: 4832
N° d'édition: 86595-207-1
N° d'ISBN: 2-86595-207-X



BASIC MSX 1. METHODES PRATIQUES

Cet ouvrage s'adresse à ceux qui, de plus en plus nombreux, ont déjà pratiqué un "BASIC" et qui souhaitent approfondir leurs connaissances informatiques sur leur ordinateur MSX.

Vous découvrirez, grâce à de nombreux programmes-exemples, toutes les instructions du Basic MSX au fur et à mesure de vos besoins, et vous utiliserez au mieux toutes les possibilités du nouveau standard : le graphique haute et basse résolution, les Sprites, la redéfinition des caractères, les sons, etc.

Les programmes de jeu et de gestion vous permettront de mettre immédiatement en pratique votre savoir théorique et de créer de fort belles pages-écran.

/ / ISBN : 2 865 95-207 X
ÉDITIONS DU P.S.I. BP 86 FRANCE - 77402 LAGNY S/MARNE CEDEX

120 FF